Sequence-to-Action: Grammatical Error Correction with Action Guided Sequence Generation

Jiquan Li[1], Junliang Guo[1], Yongxin Zhu[1], Xin Sheng[1], Deqiang Jiang[2], Bo Ren[2], Linli Xu[1]
[1]Anhui Province Key Laboratory of Big Data Analysis and Application,
School of Computer Science and Technology, University of Science and Technology of China
[2]Tencent YouTu Lab
{lijiquan, guojunll, zyx2016, xins}@mail.ustc.edu.cn, {dqiangjiang, timren}@tencent.com, linlixu@ustc.edu.cn

Abstract

The task of Grammatical Error Correction (GEC) has received remarkable attention with wide applications in Natural Language Processing (NLP) in recent years. While one of the key principles of GEC is to keep the correct parts unchanged and avoid over-correction, previous sequence-to-sequence (seq2seq) models generate results from scratch, which are not guaranteed to follow the original sentence structure and may suffer from the over-correction problem. In the meantime, the recently proposed sequence tagging models can overcome the over-correction problem by only generating edit operations, but are conditioned on human designed language-specific tagging labels. In this paper, we combine the pros and alleviate the cons of both models by proposing a novel Sequence-to-Action (S2A) module. The S2A module jointly takes the source and target sentences as input, and is able to automatically generate a token-level action sequence before predicting each token, where each action is generated from three choices named SKIP, COPY and GENerate. Then the actions are fused with the basic seq2seq framework to provide final predictions. We conduct experiments on the benchmark datasets of both English and Chinese GEC tasks. Our model consistently outperforms the seq2seq baselines, while being able to significantly alleviate the over-correction problem as well as holding better generality and diversity in the generation results compared to the sequence tagging models.

Introduction

Grammatical Error Correction (GEC), which aims at automatically correcting various kinds of errors in the given text, has received increasing attention in recent years, with wide applications in natural language processing such as post-processing the results of Automatic Speech Recognition (ASR) and Neural Machine Translation (NMT) models (Mani et al. 2020), as well as providing language assistance to non-native speakers. In the task of GEC, the primary goals are two-fold: 1) identifying as many errors as possible and successfully correcting them; 2) keeping the original correct words unchanged without bringing in new errors.

Intuitively, GEC can be considered as a machine translation task by taking the incorrect text as the source language and the corrected text as the target language. In recent years, NMT-based approaches (Bahdanau, Cho, and Bengio 2015) with sequence-to-sequence (seq2seq) architectures have become the preferred solution for the GEC task, where the original sentences with various kinds of grammatical errors are taken as the source input while the correct sentences are taken as the target supervision. Specifically, the Transformer model (Vaswani et al. 2017; Bryant et al. 2019) has been a predominant choice for NMT-based methods.

However, there are some issues with the seq2seq methods for the GEC task. In general, as a result of generating target sentences from scratch, the repetition and omission of words frequently occur in the seq2seq generation process as studied by previous works (Tu et al. 2016). More importantly, there is no guarantee that the generated sentences can keep all the original correct words while maintaining the semantic structures. Experiments show that such problems occur more frequently when the original sentences are too long or contain low-frequency/out-of-vocabulary words. Consequentially, the seq2seq models may suffer from these potential risks in practice which conflict with the second primary goal of the GEC task. Recently, sequence tagging methods (Malmi et al. 2019; Awasthi et al. 2019; Omelianchuk et al. 2020; Stahlberg and Kumar 2020) consider GEC as a text editing task by detecting and applying edits to the original sentences, therefore bypassing the above mentioned problems of seq2seq models. Nevertheless, the edits are usually constrained by human designed or automatically generated lexical rules (Omelianchuk et al. 2020; Stahlberg and Kumar 2020) and vocabularies (Awasthi et al. 2019; Malmi et al. 2019), which limits the generality and transferability of these methods. Moreover, when it comes to corrections which need longer insertions, most of these sequence tagging methods rely on iterative corrections, which can reduce the fluency.

To tackle these problems, in this paper, we propose a Sequence-to-Action (S2A) model which is able to automatically edit the erroneous tokens in the original sentences without relying on human knowledge, while keeping as many correct tokens as possible. Specifically, we simply introduce three atomic actions named `SKIP`, `COPY` and `GEN`erate to guide the model when generating the corrected outputs. By integrating the erroneous source sentence and the golden target sentence, we construct a tailored input format for our model, and for each token, the model learns to skip it if the current input token is an erroneous one, or copy it if the token is an originally correct token, or generate it if the token is a target token.

In this way, the proposed model provides the following benefits. 1) As a large proportion of tokens are correct in the source sentences, the `COPY` action will appear more frequently than the other actions. Therefore, by taking the source sentence as an input to the S2A module, our model is more inclined to copy the current input token, and therefore alleviates the over-correction problem of seq2seq models. 2) Comparing to sequence tagging models which generate edits based on the human-designed rules or vocabularies, our model can realize more flexible generations such as long insertions. More importantly, our model is language-independent with good generality.

We evaluate the S2A model on both English and Chinese GEC tasks. On the English GEC task, the proposed method achieves 52.5 in $F_{0.5}$ score, producing an improvement of 2.1 over the baseline models. Meanwhile, it achieves 38.06 in $F_{0.5}$ score on the Chinese GEC task, with improvement of 1.09 over the current state-of-the-art baseline method MaskGEC (Zhao and Wang 2020).

Related Work

GEC by Seq2seq Generation

A basic seq2seq model for GEC consists of an encoder and a decoder, where the source sentence is first encoded into the corresponding hidden states by the encoder, and each target token is then generated by the decoder conditioned on the hidden states and the previously generated tokens. Given a pair of training samples (x, y), the objective function of the seq2seq based GEC model can be written as:

$$\mathcal{L}_{\text{s2s}}(y|x;\Theta) = -\sum_{i=1}^{n} \log p(y_i|y_{<i}, x; \theta_{\text{s2s}}), \tag{1}$$

where θ_{s2s} indicates the model parameters. In general, the input and output sequences may overlap significantly for the GEC task, while a seq2seq GEC model generates the target sequences from scratch given that no source token is directly visible to the decoder, which may lead to over-correction or generation errors.

In the past few years, several methods have been proposed to improve the performance of seq2seq GEC models. Kaneko et al. (2020) incorporate a pre-trained masked language model such as BERT (Devlin et al. 2019) into a seq2seq model, which is then fine-tuned on the GEC dataset. A copy-augmented architecture is proposed in (Zhao et al. 2019) for the GEC task by copying the unchanged words from the original sentence to the target sentence. Recent advances in seq2seq GEC models mainly focus on constructing additional synthetic data for pre-training (Grundkiewicz, Junczys-Dowmunt, and Heafield 2019; Xie et al. 2018; Ge, Wei, and Zhou 2018; Kiyono et al. 2019; Zhou et al. 2020) by directly adding noise to normal sentences, through back-translation, or by using poor translation models. Nevertheless, even with the above improvements, the seq2seq GEC models still suffer from generating results from scratch, which inevitably leads to over-correction and generation errors.

GEC by Generating Edits

Another line of research takes a different perspective by treating the GEC task as a text editing problem and proposes sequence tagging models for GEC. In general, these models predict an edit tag sequence $t = (t_1, t_2, ..., t_m)$ based on the source sentence x by estimating $p(t|x) = \prod_{i=1}^{m} p(t_i|x)$. In the edit tag sequence, each tag is selected from a pre-defined set and assigned to a source token, representing an edition to this token.

Specifically, LaserTagger (Malmi et al. 2019) predicts editions between *keeping*, *deleting* or *adding* a new token/phrase from a handcrafted vocabulary. PIE (Awasthi et al. 2019) iteratively predicts token-level editions for a fixed number of iterations in an non-autoregressive way. GECToR (Omelianchuk et al. 2020) further improves the model by designing more fine-grained editions w.r.t the lexical rules of English. Seq2edits (Stahlberg and Kumar 2020) generates span-level (instead of token-level) tags to generate more compact editions. In the sequence tagging models mentioned above, the operations of generating new tokens are restricted either by human-designed vocabularies or language-dependent lexical rules, limiting their generality. For example, GECToR outperforms baseline models on English GEC datasets, but its performance severely degenerates on the Chinese GEC task, as shown in Table 3. There also exist methods that integrate seq2seq models with sequence tagging methods into a pipeline system to improve the performance or efficiency (Chen et al. 2020; Hinson, Huang, and Chen 2020), which cannot be optimized end-to-end.

Different to the works discussed above, the proposed S2A model alleviates the over-correction/omission problem of seq2seq models by taking the original sentence as input and keeping as many correct tokens as possible. Meanwhile, by generating corrections with a set of actions including `SKIP`, `COPY` and `GEN`erate which do not rely on any human-designed rules or vocabularies, S2A is language-independent and more flexible compared to text editing models.

Methodology

In this section, we elaborate on the proposed model for GEC. We start with the problem definition, followed with the motivation and the model architecture. We then proceed to demonstrate how the model is designed.

Problem Definition Given a parallel training dataset $(\mathcal{X}, \mathcal{Y})$ which consists of pairs of the original erroneous source and the golden target sequences $(x, y) \in (\mathcal{X}, \mathcal{Y})$, where $x = (x_1, x_2, ..., x_m)$ and $y = (y_1, y_2, ..., y_n)$, we aim at correcting x to y through the proposed model by optimizing $p(y|x)$.

Model Architecture

We aim to build a framework that is able to alleviate the problems of both seq2seq models and sequence tagging models as discussed in the previous section. To be specific, the model should keep the original correct tokens as much as possible, while being able to dynamically generate

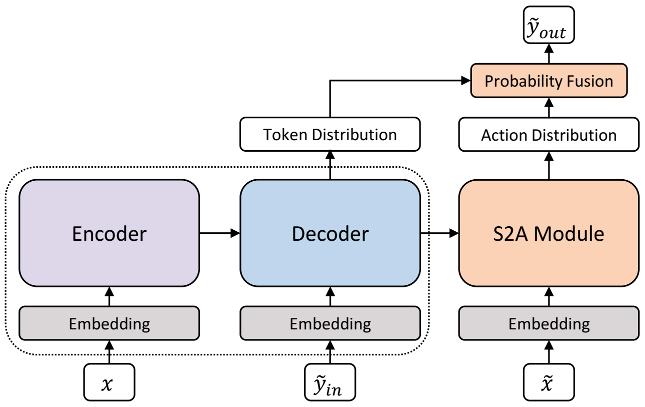

Figure 1: An illustration of the proposed framework.

editions w.r.t erroneous tokens without human knowledge. To achieve these goals, we propose a novel Sequence-to-Action (S2A) module based on the seq2seq framework, with tailored input formats. An illustration of the proposed framework is shown in Figure 1.

Specifically, our framework consists of three components: the encoder, the decoder and the S2A module. The S2A module is built upon the decoder to predict an action at each input position. Different from previous sequence tagging works which usually introduce lots of labels and generate new tokens from the pre-defined vocabulary, we simply design three actions named `SKIP`, `COPY` and `GEN`erate, where `SKIP` indicates that the current input token is an error that should not appear in the output, `COPY` refers to an original correct token that should be kept, and `GEN` indicates a new token should be generated at this position, and the token is predicted from the whole dictionary instead of a predefined subset.

The S2A module takes the hidden output of the decoder and the source sentence as input, and we design a special input format to integrate them, ensuring that at each step, the action is taken considering both the source and target information. Then the action probability produced by the S2A module is fused with the traditional seq2seq prediction probability to obtain the final results. We introduce the details as follows.

Input Construction

The encoder in our model is akin to a traditional seq2seq encoder, which takes x as input and provides its hidden representation h_e. As for the decoder side, instead of simply taking y as input, we integrate x with y as the decoder input to provide direct information of the original sentence.

Specifically, we follow four principles to integrate x and y into a new sequence z with an action sequence a. For every token in x and y, 1) if the source token is an originally correct token (i.e., appears both in x and y), we simply copy it to z and assign the action `COPY` to it; 2) if the source token is an erroneous token that should be replaced (i.e., the corresponding correct token appears in y), we jointly append this token as well as the correct token to z, and assign `SKIP` to the erroneous token while assigning `GEN` to the correct token; 3) if the source token is an erroneous token that should be deleted (i.e., no corresponding correct token appears in y), we append it to z and assign the action `SKIP` to it; 4) for the target tokens that do not have alignments in x and thus should be generated, we append them to z and assign the action `GEN`. Finally, we obtain the integrated sequence z and the action sequence a, both with length t and $t \geq \max(m, n)$. This process can be implemented by dynamic programming and we provide an illustration in Table 1.

Algorithm 1: Input Construction

Input: The source sentence x, the integrated sequence z and the action sequence a with length t
Output: The S2A input $\tilde{x}$, the decoder input $\tilde{y}_{\text{in}}$, the target sequence $\tilde{y}_{\text{out}}$

```
1:  Initialize $\tilde{x}$ and $\tilde{y}$ as empty lists; $i = 1$;
2:  for $k \leftarrow 1$ to $t$ do
3:      if $a_k ==$ COPY then
4:          Append $z_k$ to $\tilde{x}$; $i$++;
5:          Append $z_k$ to $\tilde{y}$;
6:      else if $a_k ==$ SKIP then
7:          Append $z_k$ to $\tilde{x}$; $i$++;
8:          Append BLK to $\tilde{y}$;
9:      else
10:         Append $x_i$ to $\tilde{x}$;
11:         Append $z_k$ to $\tilde{y}$;
12:     end if
13: end for
14: $\tilde{y}_{\text{out}} \leftarrow \tilde{y}$;
15: Replace BLK in $\tilde{y}$ with the token on its left;
16: $\tilde{y}_{\text{in}} \leftarrow$ [S] $+\tilde{y}_{0:-1}$;
17: return $\tilde{x}$, $\tilde{y}_{\text{in}}$, $\tilde{y}_{\text{out}}$
```

We then construct $\tilde{y}_{\text{in}}$ which serves as the decoder input and $\tilde{x}$ which maintains the source tokens and serves as the input to the S2A module. Details are described in Algorithm 1. To construct $\tilde{x}$, we iterate over z and keep the tokens with `COPY` and `SKIP` actions. For tokens with the `GEN` action, we fill up the location with its next token in the original source sentence. Meanwhile, for $\tilde{y}$, we keep the tokens with `COPY` and `GEN` actions, and fill up the locations with the `SKIP` action by introducing a special blank token `[BLK]`, which serves as a placeholder to represent the positions where the original tokens are skipped. It is worth noting that while training, we take $\tilde{y}$ as the target sequence $\tilde{y}_{\text{out}}$, and we replace the `[BLK]` token with the most recent non-blank token on its left and take the right-shifted version of $\tilde{y}$ as the decoder input $\tilde{y}_{\text{in}}$ to enable teacher forcing.

Training and Inference

Given the source input x, the decoder input $\tilde{y}_{\text{in}}$, the S2A input $\tilde{x}$ and the target sentence $\tilde{y}_{\text{out}}$, the computation flow of our model can be written as follows,

$$h = f_a(h_d, e(\tilde{x})) \quad \text{where} \quad h_d = f_d(\tilde{y}_{\text{in}}, h_e) \quad \text{and} \quad h_e = f_e(x), \tag{2}$$

	Sequence									
Source x	The	cat	is	sat	at	mat	.	[/S]		
Gold y	The	cat	sat	**on**	**the**	mat	.	[/S]		
Integrated z	The	cat	is	sat	at	**on**	**the**	mat	.	[/S]
Action a	C	C	S	C	S	**G**	**G**	C	C	C
$\tilde{x}$	The	cat	is	sat	at	mat	mat	mat	.	[/S]
$\tilde{y}_{\text{in}}$	[S]	The	cat	cat	sat	sat	**on**	**the**	mat	.
$\tilde{y}_{\text{out}}$	The	cat	[BLK]	sat	[BLK]	**on**	**the**	mat	.	[/S]

Table 1: An illustration for the decoder input construction. We refer to SKIP as S, COPY as C, and GEN as G. To differentiate the three actions, we mark SKIP and [BLK] with underlines and GEN in boldface. [S] and [/S] represent the begin- and end-of-sequence symbol.

where f_a, f_d and f_e indicate the S2A module, the decoder and encoder respectively, with the hidden outputs h, h_d and h_e correspondingly, and $e(\cdot)$ indicates the embedding lookup. Then the token prediction probability can be written as $p_d(\tilde{y}_{\text{out}}|\tilde{y}_{\text{in}}, x) = \text{softmax}(f_o(h_d))$, where $f_o(\cdot) \in \mathbb{R}^{d\times V}$ is the linear output layer that provides the prediction probability over the dictionary, while d and V indicate the dimension of the hidden outputs and the vocabulary respectively.

The S2A module simply consists of two feed-forward layers, taking the concatenation of h_d and $e(\tilde{x})$ as input and producing the probability over actions as output:

$$\begin{aligned} & p_a(a|h_d, e(\tilde{x})) \\ & = \text{softmax}(\sigma([h_d; e(\tilde{x})] \cdot w_1 + b_1) \cdot w_2 + b_2), \\ & \text{where } w_1 \in \mathbb{R}^{2d\times 2d}, w_2 \in \mathbb{R}^{2d\times 3}. \end{aligned} \tag{3}$$

For each position, the three actions SKIP, COPY and GEN indicate that the model should predict a [BLK] token, or predict the input token, or generate a new token respectively. Therefore, for the i-th position, denote the probabilities of three actions as $(p_a^i(s), p_a^i(c), p_a^i(g)) = p_a^i$, we fuse them with the token probability p_d^i to obtain the prediction probability of the S2A module $p_{\text{s2a}}^i \in \mathbb{R}^V$,

$$\begin{aligned} & p_{\text{s2a}}^i(\texttt{[BLK]}) = p_a^i(s), \\ & p_{\text{s2a}}^i(\tilde{x}^i) = p_a^i(c), \\ & p_{\text{s2a}}^i(V \setminus \{\texttt{[BLK]}, \tilde{x}^i\}) = p_a^i(g) \cdot p_d^i(V \setminus \{\texttt{[BLK]}, \tilde{x}^i\}), \end{aligned} \tag{4}$$

where $\tilde{x}^i$ indicates the current input token and $p_d^i(V \setminus \{\texttt{[BLK]}, \tilde{x}^i\})$ indicates the normalized token probability after setting the prediction of the blank token as well as the current token to 0. In this way, as the probabilities in p_a (3-classes) are usually larger than that in p_d (V-classes), with action COPY, we amplify the probability of keeping the current token which is originally correct; with action GEN, we force the model to predict a new token from the vocabulary; otherwise with action SKIP, we force the model to skip the current erroneous token by predicting the [BLK] token.

Then the loss function of the proposed Sequence-to-Action module can be written as:

$$\begin{aligned} & L_{\text{s2a}}(\tilde{y}_{\text{out}}|\tilde{y}_{\text{in}}, \tilde{x}, x) \\ & = -\sum_{i=1}^{t} \log p_{\text{s2a}}(\tilde{y}_{\text{out}}^i|\tilde{y}_{\text{in}}^{<i}, \tilde{x}^i, x; \theta_{\text{s2s}}, \theta_{\text{s2a}}), \end{aligned} \tag{5}$$

where θ_{s2a} indicates the parameters of the proposed S2A module. Together with the loss function of the traditional seq2seq model described in Equation (1), the final loss function of our framework can be written as:

$$\begin{aligned} L(x, y; \Theta) = & (1-\lambda) L_{\text{s2a}}(\tilde{y}_{\text{out}}|\tilde{y}_{\text{in}}, \tilde{x}, x; \theta_{\text{s2s}}, \theta_{\text{s2a}})) \\ & + \lambda L_{\text{s2s}}(\tilde{y}_{\text{out}}|\tilde{y}_{\text{in}}, x; \theta_{\text{s2s}}), \end{aligned} \tag{6}$$

where $\Theta = (\theta_{\text{s2s}}, \theta_{\text{s2a}})$ denotes the parameters of the model, and λ is the hyper-parameter that controls the trade-off between the two objectives.

While inference, our model generates in a similar way as the traditional seq2seq model, except that we additionally take a source token x^j as the input, which denotes the current source token that serves as the input to the S2A module. After the current generation step, if the predicted action lies in SKIP and COPY, we move the source token a step forward (i.e., j++). Otherwise with action GEN, we keep j unchanged. In this way, we ensure that x^j serves the same functionality as $\tilde{x}^i$ (i.e., the source token that should be considered at the current step) during training.

Discussion

Our model combines the advantages of both the seq2seq and the sequence tagging models, while alleviating their problems. On the one hand, while seq2seq models usually suffer from over-correction or omission of correct tokens in the source sentence (Tu et al. 2016), the proposed sequence-to-action module guarantees that the model will directly visit and consider the source tokens in $\tilde{x}$ as the next predictions' candidates before making the final predictions. On the other hand, comparing with sequence tagging models which depend on human-designed labels, lexical rules and vocabularies for generating new tokens, we only introduce three atomic actions in the model without other constraints, which enhances the generality and diversity of the generated targets. As shown in the case studies, sequence tagging models usually fail when dealing with hard editions, e.g., reordering or generating a long range of tokens, and our model performs well on these cases by generating the target sentence from scratch auto-regressively. In addition, we choose the term "action" instead of "label" because the proposed sequence-to-action module does not need any actual labels to be trained, instead it is trained end-to-end by fusing the probability with the seq2seq model.

Model	Precision	Recall	$F_{0.5}$
Transformer Big	64.9	26.6	50.4
LaserTagger (Malmi et al. 2019)	50.9	26.9	43.2
ESD+ESC (Chen et al. 2020)	**66.0**	24.7	49.5
S2A Model	65.9	**28.9**	**52.5**
Transformer Big (Ensemble)	71.3	26.5	53.3
S2A Model (Ensemble)	**72.7**	**27.6**	**54.8**
Transformer Big + pre-train	69.4	42.5	61.5
PRETLarge (Kiyono et al. 2019)	67.9	44.1	61.3
BERT-fuse (Kaneko et al. 2020)	69.2	**45.6**	62.6
Seq2Edits (Stahlberg and Kumar 2020)	63.0	**45.6**	58.6
PIE (Awasthi et al. 2019)	66.1	43.0	59.7
GECToR-BERT (Omelianchuk et al. 2020)	72.1	42.0	63.0
GECToR-XLNet (Omelianchuk et al. 2020)	**77.5**	40.1	**65.3**
ESD+ESC (Chen et al. 2020) + pre-train	72.6	37.2	61.0
S2A Model + pre-train	74.0	38.9	62.7
Transformer Big + pre-train (Ensemble)	74.1	37.5	62.0
GECToR (Ensemble)	**78.2**	**41.5**	**66.5**
S2A Model + pre-train (Ensemble)	74.9	38.4	62.9

Table 2: The results on the CoNLL-2014 English GEC task. The top group of results are generated by models trained only on the BEA-2019 training set. The bottom group of results are generated by models that are first pre-trained on a large amount of synthetic pseudo data followed with fine-tuning. Here we bold the best results of single models and ensemble models separately.

Experiments

Datasets and Evaluation Metrics

We conduct experiments on both Chinese GEC and English GEC tasks. For the Chinese GEC task, we use the dataset of NLPCC-2018 Task 2 (Zhao et al. 2018)[1], which is the first and latest benchmark dataset for Chinese GEC. Following the pre-processing settings in (Zhao and Wang 2020), we get 1.2M sentence pairs in all. Then 5k sentence pairs are randomly sampled from the whole parallel corpus as the development set. The rest are used as the final training set. We use the official test set[2] which contains 2k sentences extracted from the PKU Chinese Learner Corpus. The test set also includes the annotations that mark the golden edits of grammatical errors in each sentence. We tokenize the sentence pairs following (Zhao and Wang 2020). Specifically, we use the tokenization script of BERT[3] to tokenize the Chinese symbols and keep the non-Chinese symbols unchanged.

For English GEC, we take the datasets provided in the restricted track of the BEA-2019 GEC shared task (Bryant et al. 2019). Specifically, the training set is the concatenation of the Lang-8 corpus (Mizumoto et al. 2011), the FCE training set (Yannakoudakis, Briscoe, and Medlock 2011), NUCLE (Dahlmeier, Ng, and Wu 2013), and W&I+LOCNESS (Granger 2014; Bryant et al. 2019). We use the CoNLL-2013 (Ng et al. 2013) test set as the development set to choose the best-performing checkpoint, which is then evaluated on the benchmark test set CoNLL-2014 (Ng et al. 2014). We also conduct experiments by pre-training the model with 100M synthetic parallel examples provided by (Grundkiewicz, Junczys-Dowmunt, and Heafield 2019). All English sentences are preprocessed and tokenized by 32K SentencePiece (Kudo and Richardson 2018) Byte Pair Encoding (BPE) (Sennrich, Haddow, and Birch 2016).

We use the official MaxMatch (M^2) scorer (Dahlmeier and Ng 2012) to evaluate the models in both the Chinese and English GEC tasks. Given a source sentence and a system hypothesis, M^2 searches for a sequence of phrase-level edits between them that achieves the highest overlap with the gold-standard annotation. This optimal edit sequence is then used to calculate the values of precision, recall and $F_{0.5}$.

Model Configurations

For the basic seq2seq baseline, we adopt the original Transformer architecture. To compare with the previous works, we use the base/big model of Transformer for the Chinese/English GEC task respectively, and we follow the official settings of Transformer: in the base/big model, the number of self-attention heads of Transformer is set to $8/16$, the embedding dimension is $512/1024$, the inner-layer dimension in the feed-forward layers is set to 2048/4096 respectively. For the loss function, we use the cross entropy loss with label smoothing and set the epsilon value to 0.1. We apply dropout (Srivastava et al. 2014) on the encoders and decoders with probability of 0.3. The implementations of our model and baselines are based on `fairseq` (Ott et al. 2019) and will be available when published.

Moreover, as suggested in (Junczys-Dowmunt et al. 2018), for the English GEC task, we not only report the results generated by single models, but also compare to ensembles of four models with different initializations.

[1]http://tcci.ccf.org.cn/conference/2018/taskdata.php

[2]https://github.com/pkucoli/NLPCC2018_GEC

[3]https://github.com/google-research/bert

Model	Precision	Recall	$F_{0.5}$
YouDao (Fu, Huang, and Duan 2018)	35.24	**18.64**	29.91
AliGM (Zhou et al. 2018)	41.00	13.75	29.36
BLCU (Ren, Yang, and Xun 2018)	**41.73**	13.08	29.02
Transformer	36.57	14.27	27.86
S2A Model	36.57	18.25	**30.46**
Transformer + MaskGEC (Zhao and Wang 2020)	**44.36**	22.18	36.97
GECToR-BERT (Omelianchuk et al. 2020) + MaskGEC	41.43	23.60	35.99
S2A Model + MaskGEC	42.34	**27.11**	**38.06**

Table 3: The results on the NLPCC-2018 Chinese GEC task. The upper group of results are generated by models trained on the original NLPCC-2018 training data without data augmentation. The lower group of results are generated by models trained on the same training data but with the dynamic masking based data augmentation method proposed by MaskGEC (Zhao and Wang 2020). Here we bold the best results.

Baselines

For the English GEC task, we compare the proposed S2A model to several representative systems, including three seq2seq baselines (Transformer Big, BERT-fuse (Kaneko et al. 2020), PRETLarge (Kiyono et al. 2019)), four sequence tagging models (LaserTagger (Malmi et al. 2019), PIE (Awasthi et al. 2019), GECToR (Omelianchuk et al. 2020), Seq2Edits (Stahlberg and Kumar 2020)), and a pipeline model ESD+ESC (Chen et al. 2020). Specifically, for GECToR, we report their results when utilizing the pre-trained BERT model, XLNet model (Yang et al. 2019) and the results that integrate three different pre-trained language models in an ensemble. We denote them as GECToR-BERT, GECToR-XLNet and GECToR (Ensemble) respectively.

For the Chinese GEC task, we compare S2A to several best performing systems evaluated on the NLPCC-2018 dataset, including three top systems in the NLPCC-2018 challenge (YouDao (Fu, Huang, and Duan 2018), AliGM (Zhou et al. 2018), BLCU (Ren, Yang, and Xun 2018)), the seq2seq baseline Char Transformer, and the current state-of-the-art method MaskGEC (Zhao and Wang 2020). Note that the proposed S2A model is orthogonal to MaskGEC, and we also report our results enhanced with the data augmentation method of MaskGEC.

In addition, we reproduce and conduct Chinese GEC experiments with the sequence tagging based method GECToR (Omelianchuk et al. 2020), which is originally designed for the English task. To adapt it to the Chinese GEC task, we utilize the pre-trained Chinese BERT model `BERT-base-Chinese` and use $KEEP, $REPLACE and $APPEND as the tags. We pre-train the model using the data augmentation method proposed in MaskGEC and then fine-tune it on the training set. We follow the default hyper-parameter settings, and we set the max iteration number to 10 while inference.

Results

The English GEC results are summarized in Table 2. In the top group of results without pre-training, the proposed model achieves $52.5/54.8$ in $F_{0.5}$ score with single/ensemble model, which significantly outperforms the baselines. Compared to Transformer Big, our model achieves an improvement of 2.1 in $F_{0.5}$ score with single model, and an improvement of 1.5 in $F_{0.5}$ score with ensemble model. Further, when pre-trained on the synthetic data, our method still achieves clear gains over most of the listed models. In the meantime, it slightly outperforms BERT-fuse using the pre-trained BERT in their framework, which is not used in the S2A model. One can also notice that, as benefits of pre-trained language models, GECToR-XLNet performs the best among all the listed methods with single models, and GECToR (Ensemble) performs the best among all the baselines. Nevertheless, the proposed S2A model uses neither any pre-trained language models, nor human designed rules, and is only outperformed by the GECToR models.

We proceed with the Chinese GEC results as shown in Table 3. In the upper group, when trained on the NLPCC-2018 training set without data augmentation, our proposed model consistently outperforms all the other baseline systems. In the lower group, when trained with the data augmentation method, our model outperforms MaskGEC with an improvement of 1.09 in $F_{0.5}$ score and achieves a new state-of-the-art result.

It is worth noting that GECToR, which performs the best on the English GEC task, degenerates when generalizing to the Chinese GEC task. Without a well-designed edit vocabulary in Chinese GEC, it fails to achieve comparable performance as a standard seq2seq model even equipped with a pre-trained Chinese BERT model. In comparison, the proposed S2A framework is language independent with good generality.

Model	`COPY`	`SKIP`	`GEN`
Transformer	96.6	37.2	13.9
GECToR	**96.8**	**43.3**	12.1
S2A Model	96.7	39.8	**14.7**

Table 4: F_1 values of `COPY`, `SKIP` actions and `GEN` action fragments generated by Transformer, GECToR, S2A model on the NLPCC-2018 test set.

Type	Samples
SRC	通过些[unk][unk]，不但能回忆自己的人生，而且能准备……
TGT	通过写[unk] [unk]，不但能回忆自己的人生，而且能准备……
Transformer	我们不但能回忆自己的人生，而且能准备……
GECToR	通过写Bucket List，不但能回忆自己的人生，而且能预备……
Ours	通过写[unk] [unk]，不但能回忆自己的人生，而且能准备……
Translation	By writing [unk] [unk], not only can I recall my own life,but also prepare for ...
SRC	他就问了这个女子 就找到了他的哥哥的家。
TGT	他就问了这个女子，然后就找到了他的哥哥的家。
Transformer	他就问了这个女子 就找到了他的哥哥的家。
GECToR	他就问了这个女子 就找到了他的哥哥的家。
Ours	他就问了这个女子，然后就找到了他的哥哥的家。
Translation	He asked the woman , and then he found his brother's home.

Figure 2: Case studies of the seq2seq (Transformer) model, the sequence tagging model (GECToR) and the proposed S2A model on the Chinese NLPCC-2018 test set. The translation of the golden target in each pair is also listed. The tokens with wavy lines are errors, while tokens with underlines are the corrections made by the gold target or hypotheses. `[unk]` means out-of-vocabulary words.

Accuracy of Action Prediction

In order to analyze the impact of the S2A module, we evaluate the generated actions. To compare with other models, we extract the action sequences from the results generated by them on the NLPCC-2018 test set. Next, we calculate the F_1 values of `COPY`, `SKIP` actions and `GEN` action fragments. All scores are listed in Table 4.

As shown in Table 4, with a lower $F_{0.5}$ value of the M^2 score, GECToR performs the best in deciding whether each token should be copied or skipped from the original sentence. This is not surprising considering that GECToR is a sequence editing model. In the meantime, with an extra S2A module, our proposed framework can learn to explicitly process each token from the original sentence as GECToR does, and it performs better than Transformer in predicting `COPY` and `SKIP`. As for the action `GEN`, S2A and Transformer both generate new tokens in an auto-regressive way, which are more flexible than GECToR. As a consequence, they achieve higher accuracy when predicting the action `GEN` than GECToR, and the S2A model performs the best in this aspect.

If we look further into one of the gold-standard annotations in NLPCC 2018, among all the corrections, 1864 are replacing error characters, 130 are reordering characters, 1008 are inserting missing characters, and 789 are deleting redundant characters. Except the corrections of deleting redundant characters, the other three types of corrections all involve both deleting and extra generating. As a consequence, improving the accuracy of predicting `GEN` may help more to boost the GEC quality, which is the advantage of the proposed S2A model, and agrees with the results in Table 3.

Case Study

Next, we conduct case studies to intuitively demonstrate the advantages of the proposed S2A model. All the cases are picked from the NLPCC-2018 Chinese GEC test set. Results are listed in Figure 2.

Comparison to Seq2seq Methods In the upper part of Figure 2, we list an example in which a standard seq2seq model does not perform well. In this example, the seq2seq model suffers from word omissions. In comparison, the proposed S2A model rarely omits a long string of characters.

Comparison to Sequence Tagging Methods In the bottom example of Figure 2, a relatively long insertion is necessary which is very common in Chinese GEC. As shown in the results, the sequence tagging model is not able to produce correct output in this challenging case, while the proposed S2A model can generate the expected result due to its flexibility, which justifies the generality of S2A.

Conclusion

In this paper, we propose a Sequence-to-Action (S2A) model based on the sequence-to-sequence framework for Grammatical Error Correction. We design tailored input formats so that in each prediction step, the model learns to `COPY` the next unvisited token in the original erroneous sentence, `SKIP` it, or `GEN`erate a new token before it. The S2A model alleviates the over-correction problem, and does not rely on any human-designed rules or vocabularies, which provides a language-independent and flexible GEC framework. The superior performance on both Chinese and English GEC justifies the effectiveness and generality of the S2A framework. In the future, we will extend our idea to other text editing tasks such as text infilling, re-writing and text style transfer.

Acknowledgments

This research was supported by Anhui Provincial Natural Science Foundation (2008085J31).

References

Awasthi, A.; Sarawagi, S.; Goyal, R.; Ghosh, S.; and Piratla, V. 2019. Parallel iterative edit models for local sequence transduction. *arXiv preprint arXiv:1910.02893*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bryant, C.; Felice, M.; Andersen, Ø. E.; and Briscoe, T. 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*, 52–75. Florence, Italy: Association for Computational Linguistics.

Chen, M.; Ge, T.; Zhang, X.; Wei, F.; and Zhou, M. 2020. Improving the Efficiency of Grammatical Error Correction with Erroneous Span Detection and Correction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 7162–7169. Online: Association for Computational Linguistics.

Dahlmeier, D.; and Ng, H. T. 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*, 568–572. Montréal, Canada: Association for Computational Linguistics.

Dahlmeier, D.; Ng, H. T.; and Wu, S. M. 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*, 22–31. Atlanta, Georgia: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Fu, K.; Huang, J.; and Duan, Y. 2018. Youdao's winning solution to the nlpcc-2018 task 2 challenge: a neural machine translation approach to chinese grammatical error correction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, 341–350. Springer.

Ge, T.; Wei, F.; and Zhou, M. 2018. 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)*, 1055–1065. Melbourne, Australia: Association for Computational Linguistics.

Granger, S. 2014. The computer learner corpus: a versatile new source of data for SLA research. In *Learner English on Computer*, 3–18. Routledge.

Grundkiewicz, R.; Junczys-Dowmunt, M.; and Heafield, K. 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*, 252–263. Florence, Italy: Association for Computational Linguistics.

Hinson, C.; Huang, H.-H.; and Chen, H.-H. 2020. Heterogeneous Recycle Generation for Chinese Grammatical Error Correction. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2191–2201. Barcelona, Spain (Online): International Committee on Computational Linguistics.

Junczys-Dowmunt, M.; Grundkiewicz, R.; Guha, S.; and Heafield, K. 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)*, 595–606. New Orleans, Louisiana: Association for Computational Linguistics.

Kaneko, M.; Mita, M.; Kiyono, S.; Suzuki, J.; and Inui, K. 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*, 4248–4254. Online: Association for Computational Linguistics.

Kiyono, S.; Suzuki, J.; Mita, M.; Mizumoto, T.; and Inui, K. 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)*, 1236–1242. Hong Kong, China: Association for Computational Linguistics.

Kudo, T.; and Richardson, J. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 66–71. Brussels, Belgium: Association for Computational Linguistics.

Malmi, E.; Krause, S.; Rothe, S.; Mirylenka, D.; and Severyn, A. 2019. Encode, tag, realize: High-precision text editing. *arXiv preprint arXiv:1909.01187*.

Mani, A.; Palaskar, S.; Meripo, N. V.; Konam, S.; and Metze, F. 2020. ASR Error Correction and Domain Adaptation Using Machine Translation. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6344–6348.

Mizumoto, T.; Komachi, M.; Nagata, M.; and Matsumoto, Y. 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*, 147–155. Chiang Mai, Thailand: Asian Federation of Natural Language Processing.

Ng, H. T.; Wu, S. M.; Briscoe, T.; Hadiwinoto, C.; Susanto, R. H.; and Bryant, C. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, 1–14. Baltimore, Maryland: Association for Computational Linguistics.

Ng, H. T.; Wu, S. M.; Wu, Y.; Hadiwinoto, C.; and Tetreault, J. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, 1–12. Sofia, Bulgaria: Association for Computational Linguistics.

Omelianchuk, K.; Atrasevych, V.; Chernodub, A.; and Skurzhanskyi, O. 2020. GECToR–Grammatical Error Correction: Tag, Not Rewrite. *arXiv preprint arXiv:2005.12592*.

Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; and Auli, M. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Ren, H.; Yang, L.; and Xun, E. 2018. A sequence to sequence learning for Chinese grammatical error correction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, 401–410. Springer.

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

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958.

Stahlberg, F.; and Kumar, S. 2020. Seq2Edits: Sequence Transduction Using Span-level Edit Operations. *arXiv preprint arXiv:2009.11136*.

Tu, Z.; Lu, Z.; Liu, Y.; Liu, X.; and Li, H. 2016. Modeling Coverage for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 76–85. Berlin, Germany: Association for Computational Linguistics.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. *CoRR*, abs/1706.03762.

Xie, Z.; Genthial, G.; Xie, S.; Ng, A.; and Jurafsky, D. 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)*, 619–628. New Orleans, Louisiana: Association for Computational Linguistics.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Yannakoudakis, H.; Briscoe, T.; and Medlock, B. 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*, 180–189. Portland, Oregon, USA: Association for Computational Linguistics.

Zhao, W.; Wang, L.; Shen, K.; Jia, R.; and Liu, J. 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)*, 156–165. Minneapolis, Minnesota: Association for Computational Linguistics.

Zhao, Y.; Jiang, N.; Sun, W.; and Wan, X. 2018. Overview of the NLPCC 2018 Shared Task: Grammatical Error Correction. In Zhang, M.; Ng, V.; Zhao, D.; Li, S.; and Zan, H., eds., *Natural Language Processing and Chinese Computing*, 439–445. Cham: Springer International Publishing. ISBN 978-3-319-99501-4.

Zhao, Z.; and Wang, H. 2020. MaskGEC: Improving Neural Grammatical Error Correction via Dynamic Masking. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 1226–1233. AAAI Press.

Zhou, J.; Li, C.; Liu, H.; Bao, Z.; Xu, G.; and Li, L. 2018. Chinese grammatical error correction using statistical and neural models. In *CCF International Conference on Natural Language Processing and Chinese Computing*, 117–128. Springer.

Zhou, W.; Ge, T.; Mu, C.; Xu, K.; Wei, F.; and Zhou, M. 2020. Improving Grammatical Error Correction with Machine Translation Pairs. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 318–328. Online: Association for Computational Linguistics.