TiBERT: A Non-autoregressive Pre-trained Model for Text Editing

Baoxin Wang[1,2], Ziyue Wang[2], Wanxiang Che[1], Dayong Wu[2(✉)], Rui Zhang[3], Bo Wang[3], and Shijin Wang[2,3]

[1] Research Center for SCIR, Harbin Institute of Technology, Harbin, China
car@ir.hit.edu.cn
[2] State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, Hefei, China
{bxwang2,zywang27,dywu2,sjwang3}@iflytek.com
[3] iFLYTEK AI Research (Hebei), Langfang, China
{ruizhang19,bowang3}@iflytek.com

Abstract. Text editing refers to the task of creating new sentences by altering existing text through methods such as replacing, inserting, or deleting. Two commonly used techniques for text editing are Seq2Seq and sequence labeling. The Seq2Seq method can be time-consuming, while the sequence labeling method struggles with multi-token insertion. To solve these issues, we propose a novel pre-trained model called TiBERT, which is specially designed for **T**ext Edi**t**ing tasks. TiBERT addresses these challenges by adjusting the length of the hidden representation to insert and delete tokens. We pre-train our model using a denoising task on a large dataset. As a result, TiBERT provides not only fast inference but also an improvement in the quality of text generation. We test the model on grammatical error correction, text simplification, and Chinese spelling check tasks. The experimental results show that TiBERT predicts faster and achieves better results than other pre-trained models in these text editing tasks.

Keywords: Text Editing · Non-autoregressive Model · Pre-trained Language Model

1 Introduction

Text editing [12] is a form of text generation task, in which new sentences are created by replacing, inserting, or deleting words. The source and target sentences are often quite similar, making it appropriate to generate the target sentence by making modifications to only specific words. Typical text editing tasks include grammatical error correction (GEC) [2], text simplification (TS) [7], and Chinese spelling check (CSC) [3,8], etc.

Text editing is typically accomplished through the use of Seq2Seq and sequence labeling methods. Seq2Seq methods require the entire text to be regenerated, making them relatively slow and not fully utilizing the similarities between input and output. On the other hand, sequence labeling methods

F. Liu et al. (Eds.): NLPCC 2023, LNAI 14304, pp. 15–26, 2023.
https://doi.org/10.1007/978-3-031-44699-3_2

tend to be faster but often have difficulty handling multiple token insertions due to their limitation of inserting only one token at a time.

We propose a novel pre-trained model named TiBERT as an effective solution to enhance the performance of text editing tasks. This model is more powerful than sequence labeling methods and faster than Seq2Seq methods. Specifically, our model consists of three parts, namely, Encoder, Locator, and Editor. The Encoder is responsible for encoding the context information of the input. The Locator generates a sequence of numbers with the same length as the input. Each number of the sequence indicates the number of tokens to be generated at this position. The hidden representation of the last layer of the Encoder is edited (i.e., kept, inserted or deleted) according to the predicted editing number sequence. Then combined with a new position representation, the resulting representation is fed to the Editor. Finally, the output is generated by a non-autoregressive transformer. In this way, the problem that only one token can be added at a time for the sequence labeling method can be avoided. As shown in Fig. 1, the Locator predicts a "2" for the second input position, indicating there is one extra token to be inserted. While the "0" represents a deletion operation, meaning no token should be generated at this position.

We train our model on large-scale English and Chinese data by a denoising task. To test the effect of our model, we conduct experiments on four tasks, including English and Chinese GEC, text simplification, and CSC. The experimental results show that TiBERT runs faster and achieves better scores than other pre-trained models in all the text editing tasks.

The main contributions of this paper are as follows:

- We are the first to propose a novel pre-trained model for text editing tasks, which fills the gaps in the pre-trained model of text editing tasks.
- Our TiBERT model achieves the best results in both English and Chinese text editing tasks.
- We conduct a detailed experimental analysis and introduce application scenes for TiBERT.

2 Related Work

2.1 Text Editing Methods

Text editing methods are becoming popular solutions to natural language generation tasks with a large overlap between inputs and outputs, such as sentence fusion, style transfer, TS, and GEC. Most of these methods need to construct tag sets of editing operations before training. LaserTagger [12] and FELIX [11] employ three editing operations (the tags): token-independent *keep*, token-independent *delete* and token-dependent *add/insert*. GECToR [14] expands the tag set to 5000 token-level transformations, including basic transformations for *keep*, *delete*, *insert*, *replace*, and 29 task-specific grammatical transformations. EditNTS [7] is a two-stage method consisting of a programmer to generate an

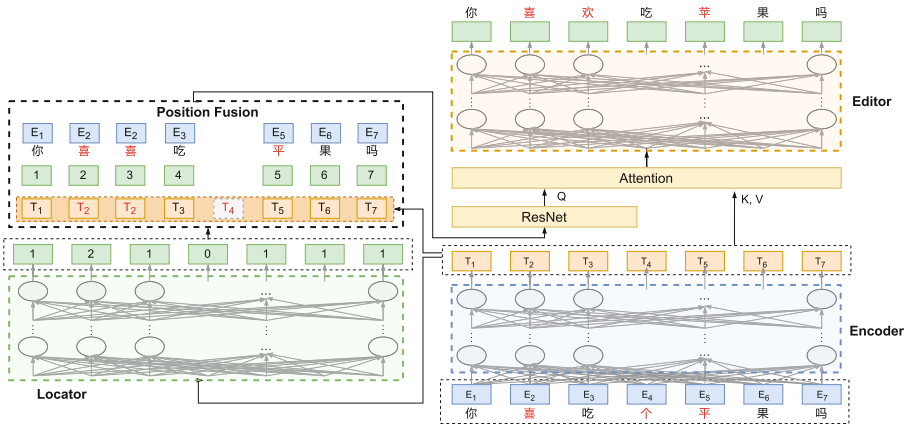

Fig. 1. The architecture of TiBERT. The input and output are translated as "Do you like apples?". The incorrect characters in the input and the corresponding corrections are shown in red. (Color figure online)

edit-operation sequence and an interpreter to recover the target text. It adds an extra operation, *stop*, to the interpreter to indicate the termination of the editing process.

Compared to other text editing models, our model can handle multi-word insertions without the need for iterative refinement. This allows our model to achieve better performance and faster prediction speed.

2.2 Pre-trained Language Models

Pre-trained language models promote the NLP tasks markedly since the presence of BERT [6]. BERT adopts the pre-training and fine-tuning mechanism. It has two pre-training tasks, next sentence prediction (NSP) and masked language model (MLM), and can be adapted to downstream tasks through task-specific fine-tuning. BERT belongs to the autoencoding model category which is better at natural language understanding (NLU) tasks such as text classification and information extraction. Contrarily, autoregressive pre-trained models, such as GPT [15] and BART [10], perform better on generation-based tasks. GPT and its improvements [16] are uni-directional models consisting of the decoder of transformers. BART includes both the encoder and the decoder. Its encoder introduces noise functions to interfere with the training data and its decoder learns to recover the original sequence. As far as we know, we are the first to pre-train a model specifically for text editing tasks.

3 Method

To enhance inference speed and tackle the challenge of inserting multiple tokens, we introduce a non-autoregressive pre-trained model, named TiBERT, to solve

the text editing task. TiBERT consists of three modules: Encoder, Locator, and Editor. The Encoder reads and comprehends the input sentences; the Locator predicts the editing number sequence to indicate the number of tokens at each position for editing; the Editor generates edited tokens according to the editing number sequence and the Encoder outputs. The overall architecture and examples of outputs of each module are illustrated in Fig. 1.

3.1 Encoder

The Encoder module is responsible for encoding the context information of the input. Similar to BERT, our Encoder employs the structure of the transformer encoder, so that our model can be trained on the basis of BERT. Moreover, the input embeddings also include position embeddings, token embeddings, and segment embeddings. The outputs of the TiBERT encoder are sent to Locator and Editor modules respectively.

$$\mathbf{H} = \text{Transformer}(\mathbf{E_t} + \mathbf{E_p} + \mathbf{E_s}) \tag{1}$$

Here, $\mathbf{E_t}$, $\mathbf{E_p}$ and $\mathbf{E_s}$ represent the token embeddings, position embeddings and segment embeddings respectively; $\mathbf{H}$ denotes the hidden representation of Encoder outputs.

3.2 Locator

The output sentences of text editing tasks are usually similar to the input sentences. Consequently, we can obtain the output by several editing operations while leaving the rest input tokens unchanged. The editing operations involve keeping, replacement, insertion, and deletion. In addition, the lengths of output and input sentences are usually unequal. In this paper, we use Locator to predict the editing number for each token from the input. The editing number is a non-negative integer, indicating the number of tokens to be generated at the corresponding location. As shown in Fig. 1, the Locator predicts the number of output tokens at each input position. Concretely, if the editing number at a position is predicted to be 0, the hidden representation at this position will not participate in the subsequent process. If the number is 3, the hidden representation of the token at that position will be extended to three copies and will participate in the subsequent operation. The equations are as follows:

$$\begin{aligned} \mathbf{H_l} &= \text{Transformer}(\mathbf{H}) \\ \mathbf{H'_l} &= \text{FFN}(\mathbf{H_l}) \\ \mathbf{P} &= \text{softmax}(\mathbf{W}\mathbf{H'_l}) \\ t_i &= \text{argmax}(p_i) \end{aligned} \tag{2}$$

where $\mathbf{H}$ is the output of Encoder, FFN is a feed-forward network used by Vaswani et al. [18]. $\mathbf{W}$ is the trainable weight; p_i is the predicted probability of the editing number at position i, and t is the editing number, indicating the number of tokens to appear at position i in the output.

3.3 Editor

The input of the Editor module consists of three parts: the hidden representations of the last layer of the Encoder, the input embeddings, and the reordered position embedding. We feed the sum of the three representations into an attention layer and get the consequential hidden representation $\mathbf{H_i}$ as follows:

$$\begin{aligned} \mathbf{H_e} &= \text{LayerNorm}(\mathbf{E'_p} + \mathbf{E'} + \mathbf{H'}) \\ \mathbf{Q} &= \mathbf{W_Q H_e}, \mathbf{K} = \mathbf{W_K H}, \mathbf{V} = \mathbf{W_V H} \\ \mathbf{H_i} &= \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \end{aligned} \tag{3}$$

where $\mathbf{E'}$ is the input embedding, which is the sum of token embedding, position embedding, and segment embedding. $\mathbf{E'_p}$ is the reordered position embedding ranging from 0 to T, T is the sum of editing numbers, $\mathbf{H'}$ is hidden representation of Encoder. $\mathbf{E'}$, $\mathbf{E'_p}$ and $\mathbf{H'}$ are transformed from $\mathbf{E}$, $\mathbf{E_p}$ and $\mathbf{H}$ respectively. Eventually, the output tokens are predicted through an n-layer transformer.

$$\begin{aligned} \mathbf{H'_i} &= \text{FFN}(\mathbf{H_i}) \\ \mathbf{H_o} &= \text{Transformer}(\mathbf{H'_i}) \end{aligned} \tag{4}$$

where $\mathbf{H_o}$ is the output representation of Editor.

3.4 Pre-training

To acquire a language model with stronger modeling and understanding ability, we pre-train TiBERT by the denoising task [10]. The denoising task requires interfering with the original sentences via extra noises and then telling the language model to denoise them. The model is trained in a more challenging manner than trained using the original data. By this means, the language modeling ability of TiBERT is enhanced. The detailed noising process is as follows:

Step 1. Input the original sentence, and randomly stream the editing number of each position from 0 to 5 according to the following probabilities, 7.5%, 80%, 7.5%, 2.5%, 2%, 0.5%, until the sum of editing numbers is greater than or equal to the length of the original sentence. If the sum grows greater than the length, we reassign the editing number of the last token to ensure that the final sum equals to the length of the original sentence. In this situation, the editing number at the last position is calculated as subtracting the sum of previous positions from the length.

Step 2. For each position, tokens are generated randomly based on the editing number. If the editing number is 0, 30% of the tokens will come from the original sentence and 70% will be randomly selected from the vocabulary. If the editing number is 1 or higher, 80% of the tokens will remain the same, 15% will be randomly selected from the vocabulary, and 5% will be taken from the original sentence.

TiBERT needs to predict the editing numbers and the editing tokens at the same time, so the final loss is a combination of the two parts, Locator and Editor. The loss function of Locator and Editor are both cross-entropy loss.

$$Loss = \lambda Loss_{locator} + (1 - \lambda) Loss_{editor} \tag{5}$$

where λ is a hyper-parameter varying from 0 to 1. For pre-training stage, we set λ to 0.5.

3.5 Fine-Tuning

After the pre-training stage, we fine-tune TiBERT with four text editing tasks. First of all, we need to convert parallel sentence pairs into editing number sequences and editing tokens. We obtain the editing numbers and the corresponding tokens at each position by Levenshtein distance. For all the editing tasks, we fine-tune our model by the loss in Equation (5). For tasks with different input and output lengths such as GEC and TS, we first generate the editing numbers by Locator and then generate the editing tokens according to the editing numbers and input tokens. For the CSC task, whose input and output lengths are the same, we input the standard editing number (all "1"s) in the test stage.

4 Experiments

We conducted experiments on various types of text editing tasks, including English and Chinese GEC, TS, and CSC. In text editing tasks, our pre-trained model works better than the autoregressive pre-trained models such as BART, and also better than the non-autoregressive models such as BERT and RoBERTa.

4.1 Settings

We use a 6-layer transformer for Encoder, a 1-layer transformer for Locator, and a 6-layer transformer for Editor. The hidden layer dimension is 768, and the intermediate size of FFN is 3072. We restrict the editing number as an integer from 0 to 5. For English pre-training, we use Colossal Cleaned CommonCrawl Corpus (C4) dataset with a total size of 305GB. For Chinese pre-training, we use Wikipedia and Wudaocorpora [25] with a total size of 152GB after data cleaning. We perform further pre-training based on BERT for English TiBERT and based on RoBERTa-wwm [4] for Chinese TiBERT. Encoder is initialized with the first 6 layers, and Editor is initialized with the last 6 layers. TiBERTs for both languages are trained with 1 million steps using batch size 2048. For the fine-tuning, 10 epochs are trained and the hyper-parameter λ is set to 0.5.

4.2 Data Conversion

Conventionally, the training data for text editing tasks are often in the form of parallel sentence pairs. Therefore, we need to convert the paired sentences into the form required for TiBERT, i.e., input token sequence, editing numbers at each position, and output token sequence. The length of output tokens and

Table 1. Experimental results on CoNLL 2014 GEC dataset. All the results are from single models.

Model	P	R	$F_{0.5}$
CopyNet [28]	65.2	33.2	54.7
PIE [1]	66.1	**43.0**	59.7
$GECToR_{BERT}$ [14]	72.1	42.0	63.0
$GECToR_{RoBERTa}$ [14]	73.9	41.5	64.0
TiBERT	**74.5**	42.1	**64.6**

Table 2. Experimental results on NLPCC 2018 GEC dataset. The best results are **bolded**, and the second best results are underlined.

Model	P	R	$F_{0.5}$
BLCU [27]	**47.63**	12.56	30.57
HRG [27]	36.79	27.82	34.56
Seq2Edit [27]	39.83	23.01	34.75
Seq2Seq [27]	37.67	**29.88**	35.80
POL-Pc [23]	46.45	23.68	38.95
TiBERT	47.21	24.86	**40.02**

the sum of editing numbers should be the same. In this paper, we convert the data format by Levenshtein, which generates a transformation between input and output. For keeping and replacing operations, the edit numbers remain 1. For the insertion operation, we add the number of inserted tokens to the origin editing number "1". For example, if adding one token to a position, the editing number of that position will be added to 2. For deletion operation, the corresponding number is 0. By the above method, we can convert the paired data form into TiBERT's input form.

4.3 Grammatical Error Correction (GEC)

The GEC task takes an erroneous sentence as the input and produces a correct version without changing the meaning. For English GEC task, we use Lang-8 [17], NUCLE [5], FCE [24] and W&I+LOCNESS[1] [2] as our training set and CoNLL 2014 as the test data. For Chinese GEC task, we conduct experiments on the training set and test set from NLPCC 2018 GEC shared task. We filter out the sentences without corrections, and use OpenCC[2] to convert all traditional characters into simplified characters. The final training data includes 1,019,371 sentence pairs. We follow the previous work and adopt $F_{0.5}$ based on MaxMatch as our evaluation method.

[1] https://www.cl.cam.ac.uk/research/nl/bea2019st/data/wi+locness_v2.1.bea19.tar.gz.

[2] https://github.com/BYVoid/.

Table 3. Experimental results on WikiLarge of Text Simplification. The best results are **bolded**, and the second best results are underlined.

Model	SARI↑	ADD↑	DELETE↑	KEEP↑	FKGL↓
PBMT-R [22]	35.92	**5.44**	32.07	70.26	10.16
NTS [13]	33.97	3.57	30.02	68.31	9.63
DRESS-LS [26]	32.98	2.57	30.77	65.60	8.94
EditNTS [7]	34.94	3.23	32.37	69.22	9.42
FELIX [11]	38.13	3.55	**40.45**	70.39	8.98
TiBERT	**38.98**	3.77	38.10	**75.07**	**8.89**

We compare our models with selected models based on Seq2Seq and sequence labeling methods. From the experimental results for English GEC task in Table 1, we can see that the performance of our model is better than that of CopyNet. TiBERT achieves 4.9% higher than that of PIE, which is a BERT-based text editing method. GECToR has designed dozens of special heuristic transformations for English grammatical error correction and achieves good results. Even so, our TiBERT still achieves better performance than the single model of GECToR based on BERT and RoBERTa.

Table 2 shows the experimental results of Chinese GEC. Seq2Edit is based on the sequence labeling method and trained from StructBERT [21]. Seq2Seq is a generation method based on BART. The effect of our model is higher than that of StructBERT and BART models, even though they are large models, whose parameters are much more than that of TiBERT. Experimental results show that our pre-trained model achieves better results than other generation methods and sequence labeling methods on the Chinese GEC task.

4.4 Text Simplification (TS)

Text simplification is a type of paraphrasing task. It reduces the content of the original text while preserving the key ideas and making it more concise. We use WikiLarge and WikiSmall [29] as our training set for the text simplification task. The test set consists of 359 source sentences taken from Wikipedia. Each source sentence contains eight references which are simplified using Amazon Mechanical Turkers. We utilize SARI and FKGL [9] as the evaluation metrics.

Table 3 shows the experimental results. EditNTS achieves good results by the editing-based method, and Felix achieves good scores based on BERT. TiBERT outperforms all the other models on the overall SARI score and the FKGL score. In addition, TiBERT performs better than FELIX on the SARI-ADD score, which implies that TiBERT has a stronger ability in adding operations.

4.5 Chinese Spelling Check (CSC)

Chinese spelling check is an important task in the field of Chinese proofreading. The numbers of input and output characters of this task are the same. We use the

Table 4. The performance on SIGHAN 2015. The best results are **bolded**, and the second best results are underlined.

Model	Detection-level			Correction-level		
	D-P	D-R	D-F	C-P	C-R	C-F
FASPell [8]	67.6	60.0	63.5	66.6	59.1	62.6
BERT [3]	73.7	78.2	75.9	70.9	75.2	73.0
SpellGCN [3]	74.8	80.7	77.7	72.1	**77.7**	74.8
RoBERTa [19]	74.7	77.3	76.0	72.1	74.5	73.3
DCN [19]	**76.6**	79.8	78.2	**74.2**	77.3	**75.7**
TiBERT	76.3	**81.8**	**79.0**	71.9	77.1	74.4

large automatically generated corpus [20][3] as our training data for CSC task. In addition, the training sets of SIGHAN 2013, SIGHAN 2014, and SIGHAN 2015 are included. We evaluate our proposed model on the test sets from SIGHAN 2015 benchmarks. Similar to the previous works, we convert the traditional characters to simplified characters by OpenCC. To compare with the state-of-the-art models, We use the widely adopted sentence-level precision, recall, and F1-score as our evaluation metrics, which have been used by Hong et al. [8][4].

We compared our model with other state-of-the-art models. As shown in Table 4, our model achieves the best performance on detection-level F1-score. TiBERT achieves 3 points higher than other pre-trained models such as BERT and RoBERTa on detection-level F1. Compared with SpellGCN and DCN models, our proposed model achieves higher detecting performance. This indicates that our model has strong detection capability in CSC tasks. However, the correction results are slightly lower than these two models'. This is because TiBERT does not use any Chinese phonetic and glyph information. As a result, TiBERT can properly detect the errors, while the predicted corrections are not the optimal answers. By contrast, SpellGCN and DCN use phonetic and glyph information to improve their performance. Even without the incorporation of additional phonetic and glyph information, TiBERT still achieves comparable performance on correction-level F1 against these models which depend on phonetic and glyph information.

5 Analysis

Generally, larger and more complex models tend to perform better. We evaluate the inference speed of several pre-trained models and find that TiBERT is slightly slower than BERT but much faster than BART. The detailed results are shown in Table 5. Additionally, our model can complete text editing tasks in a single inference, unlike other non-autoregressive models such as Levenshtein

[3] https://github.com/wdimmy/Automatic-Corpus-Generation.
[4] https://github.com/iqiyi/FASPell.

Table 5. Inference time (in ms) for BERT, BART and TiBERT on GPU (Nvidia Tesla M40). We get the average time across 100 runs.

batch size	BERT	BART	TiBERT
1	15	621	26
8	49	2480	76
32	189	5981	288

Table 6. Examples from TiBERT on text editing tasks.

Dataset	Source Sentence	TiBERT Results
CoNLL 2014	Although it looks like no laws and it is your own space to speak and do anything you want, you are actually wrong	Although it looks like **there are** no laws and it is your own space to speak and do anything you want, you are actually wrong
SIGHAN 2015	我真不好意可是今天不能参加。 Translation: I'm **sor**, but I can't attend today	我真不好**意思**是今天不能参加。 I'm **sorry**, I can't attend today

transformer and GECToR which require multiple iterations. This makes our model more efficient for text editing tasks in terms of inference speed.

We analyze the predicted results of TiBERT in the text editing task. We observe that TiBERT can effectively insert multiple tokens at a time. As shown in Table 6, for the CoNLL 2014 task, TiBERT can correctly predict that it is necessary to add two tokens and insert "there are" before "no laws" to make the sentence more fluent.

Since the SIGHAN 2015 dataset not only includes spelling errors, but also involves some extra missing errors, which may confuse TiBERT in certain situations. In Table 6, "不好意" (sor) should be changed to "不好意思" (sorry), but this will lead to the missing of "但" (but). Because SIGHAN 2015 strictly limits the consistency of input and output lengths, there are no better means to correct these two errors at the same time.

6 Conclusion

In this paper, we present a new pre-trained non-autoregressive model named TiBERT for text editing tasks. TiBERT not only guarantees the inference speed but also enhances the generation performance. It demonstrates superior performance in various text editing tasks, including GEC, text simplification, and CSC. We also conduct detailed experimental analysis and introduce application scenes for TiBERT. In the future, we will continue to explore the application of TiBERT in natural language understanding (NLU) tasks.

References

1. Awasthi, A., Sarawagi, S., Goyal, R., Ghosh, S., Piratla, V.: Parallel iterative edit models for local sequence transduction. In: Proceedings of the EMNLP-IJCNLP, pp. 4260–4270. Association for Computational Linguistics, Hong Kong, China (2019)
2. Bryant, C., Felice, M., Andersen, Ø.E., Briscoe, T.: The BEA-2019 shared task on grammatical error correction. In: Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pp. 52–75. Association for Computational Linguistics, Florence, Italy (2019)
3. Cheng, X., et al.: SpellGCN: incorporating phonological and visual similarities into language models for Chinese spelling check. In: Proceedings of the ACL, pp. 871–881. Association for Computational Linguistics, Online (2020)
4. Cui, Y., Che, W., Liu, T., Qin, B., Yang, Z.: Pre-training with whole word masking for Chinese BERT. IEEE/ACM Trans. Audio, Speech Lang. Process. **29**, 3504–3514 (2021)
5. Dahlmeier, D., Ng, H.T., Wu, S.M.: 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, pp. 22–31. Association for Computational Linguistics, Atlanta, Georgia (2013)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the NAACL, pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019)
7. Dong, Y., Li, Z., Rezagholizadeh, M., Cheung, J.C.K.: EditNTS: an neural programmer-interpreter model for sentence simplification through explicit editing. In: Proceedings of the ACL, pp. 3393–3402. Association for Computational Linguistics, Florence, Italy (2019)
8. Hong, Y., Yu, X., He, N., Liu, N., Liu, J.: FASPell: a fast, adaptable, simple, powerful Chinese spell checker based on DAE-decoder paradigm. In: Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019), pp. 160–169. Association for Computational Linguistics, Hong Kong, China (2019)
9. Kincaid, J.P., Jr, R.P.F., Rogers, R.L., Chisson, B.S.: Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel (1975)
10. Lewis, M., et al.: BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 7871–7880. Association for Computational Linguistics, Online (2020)
11. Mallinson, J., Severyn, A., Malmi, E., Garrido, G.: FELIX: flexible text editing through tagging and insertion. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 1244–1255. Association for Computational Linguistics, Online (2020)
12. Malmi, E., Krause, S., Rothe, S., Mirylenka, D., Severyn, A.: Encode, tag, realize: high-precision text editing. In: EMNLP-IJCNLP (2019)
13. Nisioi, S., Štajner, S., Ponzetto, S.P., Dinu, L.P.: Exploring neural text simplification models. In: Proceedings of the ACL, pp. 85–91. Association for Computational Linguistics, Vancouver, Canada (2017)
14. Omelianchuk, K., Atrasevych, V., Chernodub, A., Skurzhanskyi, O.: GECToR - grammatical error correction: Tag, not rewrite. In: Proceedings of the Fifteenth

Workshop on Innovative Use of NLP for Building Educational Applications,. pp. 163–170. Association for Computational Linguistics, Seattle, WA, USA Online (2020)
15. Radford, A., Narasimhan, K.: Improving language understanding by generative pre-training (2018)
16. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
17. Tajiri, T., Komachi, M., Matsumoto, Y.: Tense and aspect error correction for ESL learners using global context. In: Proceedings of the ACL, pp. 198–202. Association for Computational Linguistics, Jeju Island, Korea (2012)
18. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA, pp. 5998–6008 (2017)
19. Wang, B., Che, W., Wu, D., Wang, S., Hu, G., Liu, T.: Dynamic connected networks for chinese spelling check. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 2437–2446 (2021)
20. Wang, D., Song, Y., Li, J., Han, J., Zhang, H.: A hybrid approach to automatic corpus generation for Chinese spelling check. In: Proceedings of the EMNLP, pp. 2517–2527. Association for Computational Linguistics, Brussels, Belgium (2018)
21. Wang, W., et al.: StructBERT: incorporating language structures into pre-training for deep language understanding. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020. OpenReview.net (2020)
22. Wubben, S., van den Bosch, A., Krahmer, E.: Sentence simplification by monolingual machine translation. In: Proceedings of the ACL, pp. 1015–1024. Association for Computational Linguistics, Jeju Island, Korea (2012)
23. Xie, H., Lyu, X., Chen, X.: String editing based Chinese grammatical error diagnosis. In: Proceedings of the 29th International Conference on Computational Linguistics, pp. 5335–5344. International Committee on Computational Linguistics, Gyeongju, Republic of Korea (2022)
24. Yannakoudakis, H., Briscoe, T., Medlock, B.: A new dataset and method for automatically grading ESOL texts. In: Proceedings of the ACL, pp. 180–189. Association for Computational Linguistics, Portland, Oregon, USA (2011)
25. Yuan, S., et al.: WuDaoCorpora: a super large-scale Chinese corpora for pre-training language models. AI Open **2**, 65–68 (2021)
26. Zhang, X., Lapata, M.: Sentence simplification with deep reinforcement learning. In: Proceedings of the EMNLP, pp. 595–605. Association for Computational Linguistics (2017)
27. Zhang, Y., et al.: MuCGEC: a multi-reference multi-source evaluation dataset for Chinese grammatical error correction. In: Proceedings of NAACL-HLT. Association for Computational Linguistics, Online (2022)
28. Zhao, W., Wang, L., Shen, K., Jia, R., Liu, J.: Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In: Proceedings of the NAACL, pp. 156–165. Association for Computational Linguistics, Minneapolis, Minnesota (2019)
29. Zhu, Z., Bernhard, D., Gurevych, I.: A monolingual tree-based translation model for sentence simplification. In: Proceedings of the COLING, pp. 1353–1361. Coling 2010 Organizing Committee, Beijing, China (2010)