The BLCU System in the BEA 2019 Shared Task

Liner Yang[†], **Chencheng Wang**[‡], **Tianxin Liao**[†], **Erhong Yang**[†]
[†]Beijing Language and Culture University, Beijing, China
[‡]Beijing University of Technology, China
lineryang@gmail.com, hsamswang@gmail.com,
dubhe98@163.com, yerhong@126.com

Abstract

This paper describes the BLCU Group submissions to the Building Educational Applications (BEA) 2019 Shared Task on Grammatical Error Correction (GEC). The task is to detect and correct grammatical errors that occurred in essays. We participate in 2 tracks including the Restricted Track and the Unrestricted Track. Our system is based on a Transformer model architecture. We integrate many effective methods proposed in recent years, such as Byte Pair Encoding, model ensemble, checkpoints average and spell checker. We also corrupt the public monolingual data to further improve the performance of the model. On the test data of the BEA 2019 Shared Task, our system yields $F_{0.5}$ = 58.62 for the Restricted Track and 59.50 for the Unrestricted Track, ranking twelfth and fourth respectively.

1 Introduction

The GEC task has attracted wide interest in recent years. The goal of GEC is to detect and correct errors in essays made by English as a Second Language (ESL) learners. Since the end of both CoNLL2013 (Ng et al., 2013) and CoNLL2014 (Ng et al., 2014), many GEC researchers have used the two test sets as benchmark evaluation sets. Because of using different training sets, such as Lang-8, NUCLE, FCE, the performance of the systems are not comparable, even though they are evaluated on the same test sets. The Building Educational Applications 2019 Shared Task provides a forum for participating teams to evaluate on the same blind test set using the same training sets and evaluation metric.

Unlike previous GEC shared tasks, new corpus provided by the organizers has different CEFR[1] levels. The distribution of different levels is shown in Table 1. The training set includes essays at different levels of language ability, but no articles written by native students. There are three tracks in this shared task: Restricted Track, Unrestricted Track and Low Resource Track. Each sub-task restricts the error-corrected corpus that can be used except the Unrestricted Track. It means that the model needs to learn useful information from a large number of data written by ESL in order to correct the errors written by native learners.

	A	B	C	N	Total
Train	10,493	13,032	10,783	-	34,308
Dev	1,037	1,290	1,069	998	4,384
Test	1,107	1,330	1,010	1,030	4,477
Total	29.3%	36.3%	29.8%	4.6%	43,169

Table 1: Statistics for the sentence pairs of W&I+L corpus. A, B and C represent different CEFR levels for describing language ability, from beginner to proficient user. N denotes essays written by native English students.

In this paper, we describe the submissions from the group of Beijing Language and Culture University (BLCU) in the first two tracks. This shared task aims to tackle the full set of grammar errors, classified into 56 kinds of errors. More types of errors represent an increase in difficulty. Subtask one of the shared task (Restricted Track) restricts participants to use only the learner corpus provided by the organizers. We believe that effective use of monolingual data will enable the model to achieve better performance. Therefore, we propose a data augmentation method to corrupt a monolingual corpus with a fixed probability according to the proportion of errors in the development set and integrate many techniques proposed in recent years. We also participate in the second subtask (Unrestricted Track) which allows participants to use any learner corpus.

[1] https://www.cambridgeenglish.org/exams-and-tests/cefr/

Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 197–206
Florence, Italy, August 2, 2019.

Example	Source	Target
A	I think that **the** public transport will always **be** in the future .	I think that public transport will always **exist** in the future .
B	When the concert finished , we went to **cloakroom** to get **signatures** from musicians .	When the concert finished , we went to **the dressing room** to get **autographs** from musicians .
C	Nevertheless , **you have** another side to this reality .	Nevertheless , **there is** another side to this reality .
N	All professional boxers are at risk from being killed in **his** next fight .	All professional boxers are at risk from being killed in **their** next fight .

Table 2: Example of sentence pairs with different CEFR levels. **Bold font** represents the difference between the source and the target.

The paper is structured as follows: we present the related work in the next section. In Section 3, we describe the details of our system features. Section 4, we describe the training procedure for our system. Section 5 we specify the data sets and experiments settings. We draw our conclusions in Section 6.

2 Related Work

To our knowledge, Helping Our Own (HOO) 2011 (Dale and Kilgarriff, 2011) is the first shared task on grammatical error correction. The aim of HOO2011 is to correct errors in papers written by non-native authors (NNS), which have been published in the proceedings of ACL. Dahlmeier et al. (2011) use different open-source tools to detect spelling mistakes, prepositional errors and article errors and correct them with rule-based methods. Except for the group of University of Illinois, all participants score below 20.

Therefore, the HOO2012 shared task (Dale et al., 2012) focus more specifically on the preposition and determiner errors made by NNS who are learning English. Dahlmeier et al. (2012) treat the error correction as a classification problem and build different classifiers for determiner errors and preposition errors. Their system achieves the highest score.

The Conference on Computational Natural Language Learning (CoNLL) 2013 (Ng et al., 2013) believes that the GEC community is ready for dealing with more error types, including the two types in HOO2012, noun number, verb form and subject-verb agreement errors. Although the number of error types has increased, the most effective way at that time is still to use a pipeline of processes that combines the results from multiple systems, like Rozovskaya et al. (2013).

The CoNLL 2014 shared task (Ng et al., 2014) is the extension of CoNLL 2013, which requires participants to correct all 28 error types. Felice et al. (2014) present a hybrid approach, using statistical machine translation (SMT) as part of their pipeline system. Junczys-Dowmunt and Grundkiewicz (2014) combine Lang-8 large-scale corpus (Mizumoto et al., 2011) with the Moses (Koehn et al., 2007) SMT system. These two studies perform well, ranking first and third respectively.

Yuan and Briscoe (2016) present the first study using neural machine translation (NMT) for grammatical error correction. Xie et al. (2016a) use a character level RNN structure with attention. But all their results are worse than SMT at the same period. Chollampatt and Ng (2018) use a multilayer convolutional encoder-decoder neural network with embeddings that make use of character N-gram information. It is the first neural approach that outperforms the current state-of-the-art statistical machine translation-based approach. In the same year, Grundkiewicz and Junczys-Dowmunt (2018) combine the RNN with a phrase-based SMT system to achieve a similar score to Chollampatt and Ng (2018).

On the other hand, many scholars are committed to using additional monolingual corpora to improve the effectiveness of the models. Sennrich et al. (2016a) argue that the decoder of an NMT model is equivalent to a language model. They explore strategies to train with monolingual data without changing the NMT architecture. Yuan and Felice (2013) explore ways of generating pairs

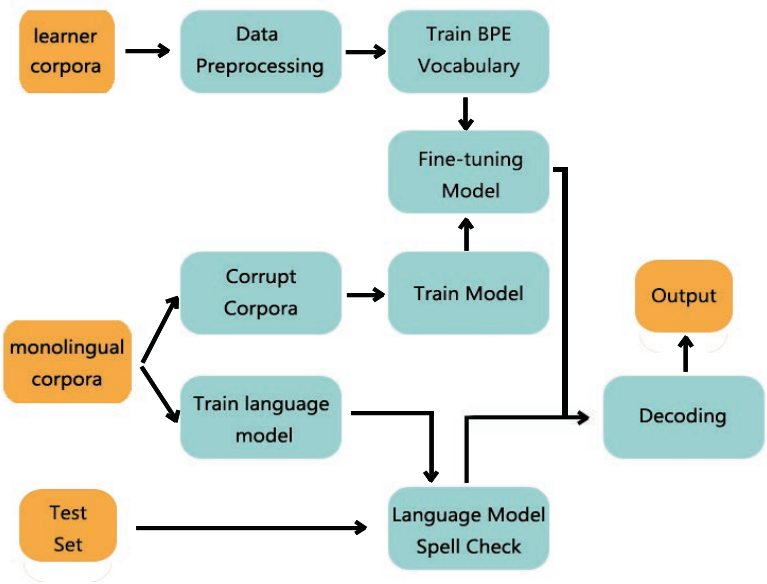

Figure 1: The pipeline of our grammatical error correction system

of incorrect and correct sentences automatically from other existing learner corpora. Both Rei et al. (2017) and Xie et al. (2018) add noise to monolingual data using Back-translation mechanism based on SMT and NMT. Wang et al. (2018) randomly replace words in both the source and target sentence with other random words from their corresponding vocabularies.

3 System Features

In this section, we will describe the features of our grammatical error correction system.

Figure 1 shows the general pipeline of the system. The training steps are shown as follows:

1) Pre-processing the learner corpora provided by the organizers.

2) Training Byte Pair Encoding (BPE) on the corpora.

3) Corrupting the One Billion Word Benchmark monolingual corpus.

4) Training model using corrupted data.

5) Fine-tuning model using the learner corpora.

The error correction steps for evaluation are:

1) Using monolingual corpus to train a language model as the spell check model.

2) Using the spell check model to correct spelling errors in the test set.

3) Decoding the output of the previous step with the grammar error correction model.

The final output after the last step forms our submission to the shared task. The following sections describe each of these components in detail.

3.1 Pre-processing and sub-words

In track one, we use four learner corpora provided by the organizers and an additional monolingual corpus.

NUCLE - This corpus is collected by the National University of Singapore and release in CoNLL shared task (Dahlmeier et al., 2013).

Lang-8 - This corpus is collected from the website called Lang-8. It is the largest publicly available learner corpus (Mizumoto et al., 2011).

FCE - The First Certificate in English corpus is collected by the University of Cambridge (Yannakoudakis et al., 2011).

W&I+L - It consists of two corpora, including Write & Improve (Bryant et al., 2019) and LOCNESS (Granger, 1998). The Write & Improve is collected by the University of Cambridge with W&I system (Yannakoudakis et al., 2018)

Source	Target
said that two Tele involved the case had been disciplined .	It said that two officers involved in the case had been disciplined .
That y to have been com their model til stat now .	That seems to have been their model up til now .
Why does sp everything have to become such a issue	Why does everything have to become such a big issue ?
Ch majority will be of the standard 6X6 configuration for carrying personnel k	The majority will be of the standard 6X6 configuration for carrying personnel .

Table 3: Example of sentence pairs made by corruption method.

and is one of the key contributions of this shared task.

One Billion - One Billion Word Benchmark dataset is a public monolingual corpus (Chelba et al., 2013), consisting of close to one billion words of English taken from news articles on the web.

All learner corpora use M2 format. For each sentence, the start and end token offsets of the wrong text range are marked and the corresponding corrections are provided.

Firstly, we extract the original sentences and modified sentences from the M2 files and write them into two files. Like in previous works (Zhao et al., 2019), we remove the unchanged sentence pairs from the original sentences and the modified sentences. We use spaCy v1.9.0[2] and the en_core_web_sm-1.2.0 model as a serializer based on the organizers' recommendations.

By analyzing the data, we find that there are many instances containing URLs in NUCLE, character encoding errors and emojis in Lang-8. So we removed the sentence pairs containing the previously mentioned case from all training sets.

In track two, we use an additional Non-public Lang-8 corpus besides all the corpora used in track one. The pre-processing method is the same as before.

In order to solve the large vocabulary and out-of-vocabulary (OOV) problem, we adopt the recommendation of Junczys-Dowmunt et al. (2018) to use the Byte Pair Encoding (BPE) algorithm (Sennrich et al., 2016b). All of the corpora are used to train the BPE vocabulary except the One Billion monolingual corpus. We split the training set, development set and the test set into sub-words using the learned BPE code. The sub-words in the test set will be merged before evaluating.

[2] https://spacy.io/

3.2 Corrupting Corpora

Many resent work regard grammatical error correction as a low-resource neural machine translation task (Junczys-Dowmunt et al., 2018; Zhao et al., 2019; Lichtarge et al., 2019). Both Grundkiewicz and Junczys-Dowmunt (2014) and Lichtarge et al. (2019) use the Wikipedia revision histories to generate additional corpora. Junczys-Dowmunt and Grundkiewicz (2016); Junczys-Dowmunt et al. (2018) utilize the Common Crawl corpus to train the language model and pre-train part of the NMT model. Inspired by these studies, we also try to use a monolingual corpus for data augmentation.

First, we define the error rate of the corpus as:

$$Er(C) = \frac{1}{n}\sum_{i=0}^{n}\frac{\text{levenshtein}(src, trg)}{\text{length}(trg)} \quad (1)$$

where n is the number of sentence pairs in corpus, src refers to the sentence to be modified, trg is the modification of src. levenshtein(src, trg) means the shortest edit distance for the src and trg in terms of tokens. length(trg) refers to the number of token in trg.

Secondly, we assume that errors in the corpus can be divided into three types (Bryant et al., 2017):

- Missing type (**M**)

  src : w_0 ... w_{i-1} w_{i+1} ... w_n

  trg : w_0 ... w_{i-1} w_i w_{i+1} ... w_n

  Token w_i is a missing type.

- Unnecessary type (**U**)

  src : w_0 ... w_{i-1} w_i w_{i+1} ... w_n

  trg : w_0 ... w_{i-1} w_{i+1} ... w_n

  Token w_i is an unnecessary type.

- Replacement type (**R**)

  src : w_0 ... w_{i-1} $w_{i'}$ w_{i+1} ... w_n

  trg : w_0 ... w_{i-1} w_i w_{i+1} ... w_n

  Token w_i is a replacement type.

Counting all of the training sets, we find that the error rate is 30% and the ratio of M : U : R is 1:1:1. We apply this to corrupt the monolingual corpus. It means that 30% tokens in the training set will be corrupted. The steps of corruption are shown as follows:

- Delete the token with a probability of 33%.
- Randomly add a token in the vocabulary with a probability of 33%.
- Randomly replace a token in the vocabulary with a probability of 34%.

This process produces a large number of wrong sentences. Finally, the original One Billion Word Benchmark corpus sentence is the target sentence and the output of the corruption system is the corresponding source sentence.

3.3 Transformer

As mentioned in the previous section, neural machine translation has become the state-of-the-art approach for Grammatical Error Correction. We adopt the attention-based NMT model proposed by Vaswani (Vaswani et al., 2017).

The embedding layer is divided into two embeddings, including token and position embedding. The token embedding contains the vector corresponding to each token, and the position embedding contains the vector of each absolute position. The embedding layer encodes each token S_i^{src} of the input sentence S^{src} into a vector $h_{S_i}^{src}$ by looking up in a token embedding matrix and adding a position vector, as shown in Eq (2).

$$h_{S_i}^{src} = Token\_emb(S_i^{src}) + Position\_emb(i) \quad (2)$$

The Encoder has N identical attention blocks, each block containing a Multi-Head attention and a linear layer. The Multi-Head is the concatenation of the N attention heads. The Encoder produces the input context-aware hidden state, shown in the Eq (3,4).

$$MultiHead(h_S^{src}, h_S^{src}, h_S^{src}) = Concat(Attention(h_S^{src}, h_S^{src}, h_S^{src})) \quad (3)$$

$$HS_S^{src} = Encoder(h_S^{src}) \quad (4)$$

The structure of the decoder is similar to that of the encoder, with N identical attention blocks. The only difference is that the decoder's attention block has an extra Multi-Head attention which attends over the encoder's context-aware hidden state. The decoder updates the hidden state of the current layer based on the attention output from the encoder and the hidden states of previous layer:

$$HS_{S_{i+1}}^{trg} = Decoder(HS_S^{src}, HS_{S_i}^{trg}) \quad (5)$$

The final decoder layer output vector HS_S^{trg} is dot-multiplied with the output embedding. Applying softmax on the inner product's output can get the predicted probability of each word, like Eq (6). Words with the highest predicted probability are chosen as the final output.

$$p(S_{i+1}|S_1, ..., S_i, S) = softmax(OutEmbedding(HS_S^{trg})) \quad (6)$$

The model can be trained with maximum likelihood estimation, as shown in Eq(7):

$$L(S^{trg}) = -\sum_{i=1}^{T} log(p(S_i^{trg})) \quad (7)$$

For the grammatical error correction task, the model copy correct tokens in most cases. But what the model really needs to learn is to translate the wrong tokens into the right ones. Therefore, we add the Edit-weighted MLE objective (Junczys-Dowmunt et al., 2018) into the loss function to give the wrong tokens greater penalty. Details of the implementation are shown as follows:

$$L(S^{trg}) = -\sum_{i=1}^{T} A(S_j^{src}, S_i^{trg}) log(p(S_i^{trg}))$$
$$A(S_j^{src}, S_i^{trg}) = \begin{cases} \Lambda \ if \ S_j^{src} \neq S_i^{trg} \\ 1 \ otherwise \end{cases} \quad (8)$$

where $A(S_j^{src}, S_i^{trg})$ is an alignment between source token and target token. It means that if the source token is inconsistent with the target, the loss value will be multiplied by Λ.

3.4 Language Model based Spell Checker

As mentioned in Xie et al. (2016b) and Chollampatt and Ng (2017), a token-based neural machine translation model is not designed to correct spelling mistakes. To address this issue, we have adopted a language model based spell checker.

We use Kenlm[3] to build a 5-gram language model from the Billion Word Benchmark dataset. Based on this language model, we use CyHunspell[4] which is a Python wrapper for Hunspell to correct spelling errors in corpora.

Algorithm 1 describes the process of correcting the corpus using the spell check model.

Algorithm 1 Language Model based Spell Checker

Input: Language Model LM, corpus with errors E, error correction threshold η, CyHunspell spell checker $Spell$
Output: corrected corpus C

```
Initialize C = {}
for all sentences s_i in E do
    score = LM.score(s_i)
    for all tokens t_j in s_i do
        candidate = Spell.suggest(t_j)
        temp = s_i.apply(candidate)
        tempscore = LM.score(temp)
        if tempscore/score > η then
            s_i = s_i.apply(candidate)
        end if
    end for
    Add s_i to C
end for
return C
```

For each sentence in E, record the language model score. The modification will be applied only if the ratio of $tempscore$ to $score$ is greater than η. Finally, the output of the program is the corrected result.

4 Training procedure

Augmentation data for corruption is collected from articles on news sites. The Lang-8 corpus used in training is written by many second language learners about their daily life. Corrupted corpus and learner corpus belong to different domains. Moreover, the errors contained in the augmentation data are not common errors for second language learners.

Based on this situation, we train on the augmentation corpus and the learner corpus separately. Firstly, we pre-train the corrupted corpus for 5 epochs. We use the arithmetic mean of the last three epochs as the final weighed result of the pre-training.

Next, we fine-tune the pre-trained model using the learner corpus consisting of (Non-public) Lang-8, NUCLE, FCE and W&I+L datasets and evaluate on the development set at each end of epoch. For each single model, we calculate the arithmetic mean of the five epochs with the best cross-entropy cost on the development set as the final model.

Our model is composed of the ensemble of 8 single models. The hyper-parameters and the training procedure used in each single model are the same except the random seed.

5 Data sets and Experiments

In this section, we will detail the data sets, the hyper-parameters and the open source tools we use.

5.1 Data sets

The statistics for the data we use in this shared task are shown in Table 6. We implement the pre-processing method mentioned in Section 3.1 for both tracks. The first four rows list the fine-tuning datasets we use in track one. The fifth line summaries the above datasets. The Non-public Lang-8 in the sixth line is the additional corpus we collect from Lang-8[5]. It is worth mentioning that some instances of Non-public Lang-8 also exist in Lang-8. We use the union of all learner corpora as the training data for track two, including 6 million

[3] https://github.com/kpu/kenlm
[4] https://pypi.org/project/CyHunspell/
[5] https://lang-8.com/

Model	without LM Spell Checker			with LM Spell Checker		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
transformer single	43.11	24.98	37.65	43.61	26.87	38.78
transformer single + CC	46.14	27.66	40.71	46.23	29.50	41.52
transformer ensemble + CC	48.83	26.39	41.79	48.96	28.29	42.72

Table 4: The evaluation of our system on the track one development set. Transformer single refers to the single model, while transformer ensemble denotes the ensemble of 8 single models. CC means use additional corrupted corpus.

	Restricted Track			Unrestricted Track		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
A	64.26	49.89	60.76	68.29	54.18	64.91
B	61.83	48.71	58.67	61.89	53.87	60.10
C	60.75	55.84	59.70	56.31	64.04	57.71
N	49.02	58.86	50.71	44.25	62.58	47.00
ALL	60.81	51.22	58.62	60.32	56.42	59.50

Table 5: The evaluation of our system on the test set.

Corpus	Before process	After process
NUCLE	56,670	21,242
FCE	32,844	20,552
Lang-8	1,112,513	560,542
W&I+L	34,308	22,544
Track One	1,236,335	624,880
Non-public Lang-8	8,655,173	6,230,606
Track Two	9,891,508	6,456,889
One Billion*	30,301,028	20,032,188
W&I+L(dev)	4,384	-
W&I+L(test)*	4,477	-

Table 6: Number of the sentence pair for different dataset. Track one summaries the statistic of all of the data we use in track one, and so does Track Two. Non-public Lang-8 is the additional corpus we use in track two. W&I+L(dev,test) is provided by the organizers. * indicates that this corpus has no target available.

sentence pairs. The last two rows in the table show the size of the development and test set.

5.2 Experimental settings

In this shared task, we use the Transformer model (Vaswani et al., 2017) implemented by FAIR[6] as the GEC model. The detailed parameters of the model are as follows: model BPE embeddings are trained for 50,000 steps (Junczys-Dowmunt et al., 2018) on the error-annotated data by the subword algorithm[7]. Both the source embedding and the target embeddings have 512 dimensions and use the same vocabulary. We share the weights of decoder input and output embeddings. Both of the encoder and decoder have 6 multi-head layers and 8 attention heads. The size of the inner layer at each layer is 2048.

We use the Adam optimizer (Kingma and Ba, 2014) to train transformer with inverse squared root schedule which decays the learning rate based on the inverse square root of the warm-up steps. The initial learning rate is 5×10^{-4} and the warm-up step is set to be 4000. We use a batch size of approximately 32,000 tokens and fine-tune the model on learner corpus for 50,000 steps. Dropout is applied at a ratio of 0.3. The loss factor Λ is set to 1.2.

The ensemble model is composed of 8 identical Transformers trained and fine-tuned separately. The only difference between them is that they use different random seeds.

During model inference, we run beam search with the emsemble model and set the beam size to 12. We use ERRANT[8] (Bryant et al., 2017) to evaluate the decoding results.

5.3 Experiment result and analysis

Table 4 shows the performance of our model with different settings. For the *without LM Spell Checker* columns, we do not use language model based spell checker to correct spelling mistakes,

[6] https://github.com/pytorch/fairseq

[7] https://github.com/rsennrich/subword-nmt

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

while *with LM Spell Checker* is the opposite. The first two lines report the result with a single Transformer model, and the last line with the ensemble model. + *CC* means that we pretrain the transformer model using the corrupted corpus and then fine-tune with learner corpus. We submit the best model, namely the ensemble model, for the shared task.

In Table 4, we can see that the main contribution comes from the corruption method. About 20 million monolingual data have brought about an increase of 3.06 in terms of F-measure on a single model. The spell checker based on the language model improves the performance of the model by about one point. We use an ensemble of identical models (except for the random seed), but we will attempt to use different types of models in future work.

Table 5 shows the result on the test set which is evaluated by the organizers. Comparing the results of the two tracks, we find that training with the Non-public Lang-8 data can significantly improve the recall about 5-10 points. However, in terms of $F_{0.5}$, the performance of the model has only been significantly improved on the test data at A and B levels, and has dropped by about two points in C and N. One possible explanation is that the errors contained in the Non-public Lang-8 belong to the lower CEFR level. Overtraining in a large amount of data containing beginner errors has reduced the performance of our system at C and N levels.

6 Conclusions

In this paper, we have described the submission to the BEA 2019 shared task on Grammatical Error Correction. Our approach combines a method of data augmentation with a pipeline system based on the Transformer model. We first corrupt the monolingual corpus and pre-train a single model on it. Then we fine-tune on the learner corpora and ensemble eight single Transformer models to further improve the performance.

The results of our best system on the blind test set are $F_{0.5}$ = 58.62 for the Restricted Track and $F_{0.5}$ = 59.50 for the Unrestricted Track, placing our system in the twelfth and fourth place respectively.

Acknowledgments

This research project is supported by Science Foundation of Beijing Language and Culture University (supported by "the Fundamental Research Funds for the Central Universities") (18YBB20), and the National Natural Science Foundation of China (61872402).

References

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 Shared Task on Grammatical Error Correction. In *Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: Towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*.

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

Daniel Dahlmeier, Hwee Tou Ng, and Eric Jun Feng Ng. 2012. Nus at the hoo 2012 shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*.

Daniel Dahlmeier, Hwee Tou Ng, and Thanh Phu Tran. 2011. Nus at the hoo 2011 pilot shared task. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics.

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*.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. Hoo 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The hoo 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics.

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

Sylviane Granger. 1998. The computer learner corpus: A versatile new source of data for SLA research. In Sylviane Granger, editor, *Learner English on Computer*, pages 3–18. Addison Wesley Longman, London and New York.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2014. The wiked error corpus: A corpus of corrective wikipedia edits and its application to grammatical error correction. In *International Conference on Natural Language Processing*. Springer.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The amu system in the conll-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. *arXiv preprint arXiv:1605.06353*.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. *arXiv preprint arXiv:1904.05780*.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*.

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*. Association for Computational Linguistics.

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

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

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, and Dan Roth. 2013. The university of illinois system in the conll-2013 shared task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016a. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016b. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Helen Yannakoudakis, Øistein E Andersen, Ardeshir Geranpayeh, Ted Briscoe, and Diane Nicholls. 2018. Developing an automated writing placement system for esl learners. *Applied Measurement in Education*, 31(3):251–267.

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*. Association for Computational Linguistics.

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

Zheng Yuan and Mariano Felice. 2013. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.