Heterogeneous Recycle Generation for Chinese Grammatical Error Correction

Charles Hinson,[1] Hen-Hsen Huang,[2,3] and Hsin-Hsi Chen[1,3]
[1] Department of Computer Science and Information Engineering,
National Taiwan University, Taiwan
[2] Department of Computer Science, National Chengchi University, Taiwan
[3] MOST Joint Research Center for AI Technology and All Vista Healthcare, Taiwan
charles.hinson@nlg.csie.ntu.edu.tw, hhhuang@nccu.edu.tw,
hhchen@ntu.edu.tw

Abstract

Most recent works in the field of grammatical error correction (GEC) rely on neural machine translation-based models. Although these models boast impressive performance, they require a massive amount of data to properly train. Furthermore, NMT-based systems treat GEC purely as a translation task and overlook the editing aspect of it. In this work we propose a heterogeneous approach to Chinese GEC, composed of a NMT-based model, a sequence editing model, and a spell checker. Our methodology not only achieves a new state-of-the-art performance for Chinese GEC, but also does so without relying on data augmentation or GEC-specific architecture changes. We further experiment with all possible configurations of our system with respect to model composition order and number of rounds of correction. A detailed analysis of each model and their contributions to the correction process is performed by adapting the ERRANT scorer to be able to score Chinese sentences.

1 Introduction

Grammatical error correction (GEC) is the task of correcting grammatical and spelling errors that appear in a sentence. An example of Chinese GEC is correcting the word-choice error in the following sentence:

本人是在貴公司的一名實習。 (I am an internship at your company.)

by changing the word 實習 (internship) to 實習生 (intern), resulting in the corrected sentence:

本人是在貴公司的一名實習生。 (I am an intern at your company.)

In recent years, there has been a great deal of GEC related research for English, most notably with the CoNLL 2014 shared task (Ng et al., 2014) and the BEA-2019 shared task (Bryant et al., 2019). Chinese GEC has a much shorter history, with the NLPCC 2018 shared task (Zhao et al., 2018) being the first to focus on this research topic. Most work prior to the NLPCC 2018 shared task focused on correcting only one type of error, such as preposition errors (Huang et al., 2016) or Chinese spelling error correction (Wu et al., 2013).

Most recent work in GEC formulate correction as a translation task, and use neural machine translation (NMT) based models. That is, models are trained to *translate* an erroneous source sentence into a corrected target sentence. A considerable disadvantage of this approach is that NMT-based systems require an enormous amount of training data to achieve good results, while the availability of parallel correction data is limited in many languages. The current leading methods for English GEC both rely on pre-training models with a large amount of artificially generated data (Grundkiewicz et al., 2019; Kiyono et al., 2019). In this work, we aim to avoid this issue by combining several different models that perform corrections in different ways.

Another challenge of GEC is that sentences can have multiple errors. Sometimes a model is not able to correct all of the errors present in a sentence in one pass, resulting in only a partial correction. One of the methods used to resolve this issue is *recycle generation*, also known as *iterative decoding* (Lichtarge et al., 2018). In this method, a system performs multiple iterations of correction on an erroneous sentence.

Proceedings of the 28th International Conference on Computational Linguistics, pages 2191–2201
Barcelona, Spain (Online), December 8-13, 2020

The current state-of-the-art method for Chinese GEC (Qiu and Qu, 2019) also employs recycle generation to solve this issue. A drawback of current recycle generation methods is that for each round of correction the same model is re-applied, resulting in a limited coverage of error corrections. We postulate that larger performance gains can be had if *different* kinds of models perform each round of correction. In this work, we propose applying recycle generation to Chinese GEC using a heterogeneous system composed of different kinds of models, covering more diverse error types.

Our idea is to leverage the advantages of both machine translation models and sequence editing editing models. Machine translation models are capable of rewriting the entire sentence, making large scale corrections such as re-ordering or performing multi-word substitutions possible. In contrast, sequence editing models focus on smaller scale corrections, such as removing a word or adding in a missing punctuation mark. Each family of model is adept at correcting different kinds of errors. By integrating these two kinds of models using recycle generation, a wider range of errors can be effectively corrected for each round of correction.

In this work, we also discuss the performance metrics of Chinese GEC. The Maxmatch (M2) scorer (Dahlmeier and Ng, 2012) that has been extensively used for English GEC, and also for the NLPCC 2018 Chinese GEC task can only report overall model performance. This problem has been solved in English GEC, with the introduction of the ERRANT scorer (Bryant et al., 2017). The ERRANT scorer can provide model performance in terms of edit-level operation as well as specific English grammatical error types. We extend the idea of the ERRANT scorer to deal with Chinese sentences. This will allow Chinese GEC researchers to be able to get more detailed analysis of model performance. In summary, our contributions are threefold as follows.

1. We use a heterogeneous system composed of multiple kinds of models for Chinese GEC, beating the previous state-of-the-art results on the NLPCC 2018 task dataset. Combining multiple models that are designed to correct different kinds of errors enables this method to achieve good results without a vast amount of training data.

2. We experiment with recycle generation to find the optimal model composition order and number of correction iterations for our system.

3. We adapt the ERRANT scorer to be able to annotate and score Chinese sentences, allowing us and future researchers for Chinese GEC to be able to get more detailed model performance results.

2 Related Work

2.1 NMT-based Methods

Current state-of-the-art results for English GEC use sequence to sequence Transformers (Vaswani et al., 2017), and rely on pre-training with large amounts of artificial data. Grundkiewicz et al. (2019) use a rule based method, leveraging confusion sets to generate 100 million sentence pairs to use as pre-training data. Kiyono et al. (2019) experiment with several variants of backtranslation (Sennrich et al., 2015) using different monolingual seed corpora to generate 70 million artificial sentence pairs.

Systems for Chinese GEC also rely on sequence to sequence models. The NLPCC 2018 shared task winner uses five different models in tandem, and chooses the best output with a 5-gram language model (Fu et al., 2018). Ren et al. (2018) use an ensemble of Convolutional sequence to sequence models with pre-trained word embeddings. The current state-of-the-art system proposed by Qiu and Qu (2019) uses a Transformer sequence to sequence model with heterogeneous recycle generation and a spellchecker.

2.2 Sequence Editing Models

Sequence editing models, also known as a *text-editing model*, learn to edit a sequence through applying a fixed set of operations to the input. This can be formulated in the following way: given a fixed vocabulary of edit operations E and an input sequence $x_{1:n} = (x_1, \cdots, x_n)$, a model learns to predict an edit operation $e_i \in E$ for each x_i in our input sequence. A set of rules can then be applied to the output sequence $e_{1:n}$ to obtain the target output sequence y.

Several sequence editing models have been proposed for text simplification tasks. The Levenshtein Transformer (Gu et al., 2019) performs text simplification by using a sequence of insertion and deletion operations. Dong et al. (2019) perform sequence editing through three primary edit operations, KEEP, ADD, and DELETE.

Currently, the only sequence editing model to be applied to GEC is LaserTagger (Malmi et al., 2019). Similarly to the two previously cited works, LaserTagger learns to edit sentences by two different edit operations: KEEP and DELETE, along with pairing these operations with a limited phrase vocabulary consisting of tokens that are frequently changed between the source and target sequences. While LaserTagger performed well for English GEC considering the small number of training samples that it used, it was still very far from reaching state-of-the-art performance. In this work, we apply LaserTagger to Chinese GEC, and also explore combining it with NMT-based models.

2.3 Recycle Generation

Recycle generation refers to the method of performing multiple rounds of correction on an input sentence. Recycle generation is also known as *iterative decoding* or *multi-pass decoding*. This has been attempted in English GEC in which one NMT-based model is used repeatedly (Lichtarge et al., 2018), or with a combination of a SMT-based and NMT-based model (Grundkiewicz and Junczys-Dowmunt, 2018). In Chinese GEC, only NMT-based recycle generation has been used (Qiu and Qu, 2019). In previous works, recycle generation has always been performed with models trained to do translation. In this work, we attempt to perform recycle generation with one model trained to do translation and another model trained to do sequence editing.

3 Methodology

Our GEC system is composed of three separate components: a neural machine translation system, a sequence editing system, and a spell-checker. Each model performs one or several rounds of correction on the input sentence to produce the final corrected output. Each component has separate strengths and weaknesses in terms of the errors it can correct. Composing the different models in this way allows **(1)** one model to correct any errors that another model has missed and **(2)** a model can fix any errors accidentally introduced by another model earlier in the pipeline. An example of the entire correction process applied to a sentence can be seen in Table 1.

Model	Input	Output	Change
NMT	我在家哩一个人学习中文。	我在家哩自学习中文。	一个人 → 自
SE	我在家哩自学习中文。	[K][K][K][K][K][K][D][K][K][K]	Delete 习
SC	我在家哩自学中文。	我在家里自学中文。	哩 → 里

Table 1: Example of full correction pipeline. The input to each model is a Chinese sentence. For the sequence editing model [K] stands for KEEP and [D] stands for delete.

3.1 Datasets and preprocessing

We use the NLPCC 2018 shared task dataset (Zhao et al., 2018) for our experiments. The training set consists of 717,241 sentences from lang8,[1] and the test set consists of 2,000 sentences from the PKU Chinese Learner Corpus. Each sentence in the test set is corrected by two annotators, and also labeled with error type information. Each error is categorized into one of four types: redundant (R), missing (M), word selection errors (S), and word ordering errors (W). Many samples from the lang8 training set have multiple alternative corrections. We expand each alternative correction into a separate sample. This process results in a training set of 1,222,906 correction pairs. Since an official validation set is not provided, we randomly select 5,000 pairs from the training set to serve as a validation set. In addition to

[1] https://lang-8.com/

samples from lang8, we also use monolingual WMT News data (Barrault et al., 2019) to form a training set for the language model that we use in our spell checker.

All three models that we use require different preprocessing steps, due to several different reasons. The NMT-based model uses standard preprocessing steps: word-level tokenization followed by subword segmentation of rare words using byte pair encoding (Sennrich et al., 2016) to handle out of vocabulary words. Word-level segmentation is performed using Jieba.[2] BPE is performed using subword-nmt,[3] with the number of merge operations set to 35k and the vocabulary threshold set to 50. The language model cannot use subword units because the spelling check algorithm that we use requires looking up words in a dictionary. The sequence editing model has better results with character-level segmentation because the algorithm it uses to build its vocabulary is sensitive to any noise introduced by incorrect segmentation that often occurs in the erroneous source sentences. For all models, we use OpenCC[4] to convert traditional Chinese characters in the training and validation sets to simplified Chinese characters. An overview of the dataset splits and preprocessing steps required for each model can be seen in Table 2. Table 3 shows an example of the different preprocessing steps applied to a sentence.

Corpus	Sentences	Split	Models
lang8	1,215,906	train	S2S, LT, LM
lang8	5,000	valid	S2S, LT, LM
PKU	2,000	test	S2S, LT, LM
WMT News	4,724,008	train	LM

(a) Datasets and splits

Model	Preprocessing
S2S	OpenCC + Jieba + BPE
LT	OpenCC
LM	OpenCC + Jieba

(b) Preprocessing steps per model

Table 2: Overview of datasets and preprocessing steps for each model. S2S stands for Neural Machine Translation model, LT stands for LaserTagger, and LM stands for language model.

Sentence	他們有兩個孩子，一男一女
English	They have two children, one boy one girl.
OpenCC	他們有兩個孩子，一男一女
Jieba	他們 有 兩個 孩子 ， 一男一女
BPE	他們 有 兩個 孩子 ， 一@@ 男@@ 一@@ 女

Table 3: Example of preprocessing steps.

3.2 Neural Machine Translation Model

Our NMT model, based on the Transformer architecture (Vaswani et al., 2017), is an encoder-decoder sequence to sequence model, where both the encoder and decoder are composed of six layers of self-attention modules. We use the "Transformer (big)" settings described in Vaswani et al. (2017). In general, we follow similar training steps as described in English state-of-the-art models (Kiyono et al., 2019; Grundkiewicz et al., 2019).

Training Settings Our model is implemented using the Fairseq[5] toolkit (Ott et al., 2019). Optimization is performed using the Adam (Kingma and Ba, 2014) optimizer, with criterion set to label-smoothed cross entropy (Szegedy et al., 2016). We use beta values of 0.9 and 0.98 for Adam, and a smoothing value of 0.1 for the criterion. We first set the learning rate to 10^{-7} and perform 4,000 warm-up updates. After the warm-up period, the learning rate is increased to 0.001. Thereafter we use an inverse square

[2] https://github.com/fxsjy/jieba
[3] https://github.com/rsennrich/subword-nmt
[4] https://github.com/BYVoid/OpenCC
[5] https://github.com/pytorch/fairseq

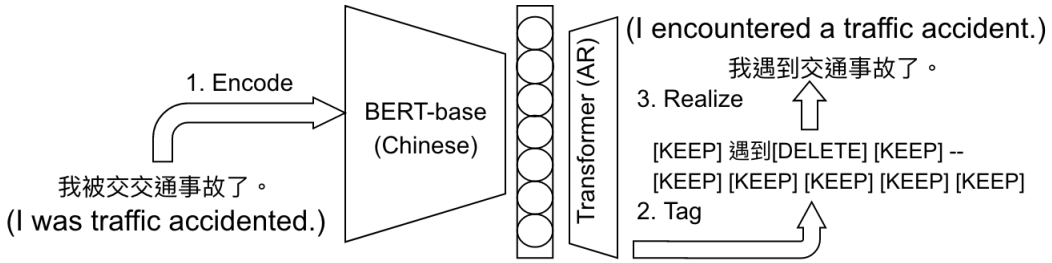


Figure 1: An example of LaserTagger applied to a Chinese sentence

root scheduler to decay the learning rate in proportion to the number of updates. Dropout between Transformer layers is set to 0.3, and attention layers is set to 0.1. We share weights between input and output embeddings. The batch size is set to a maximum of 8,484 tokens per batch. The model is trained for 40 epochs, with a checkpoint being saved at every epoch. Weights of the last 7 checkpoints are averaged together to create our final model.

3.3 Sequence Editing Model

For sequence editing, we adapt LaserTagger (Malmi et al., 2019) to be used for Chinese sentences. As described in Malmi et al. (2019), a sequence editing model learns to generate a target sentence by applying a small set of edit operations to the source sentence. It works in three steps: **(1)** the input sentence is encoded into a hidden representation, **(2)** each token in the input sentence is assigned an edit tag, and **(3)** rules are applied to convert the output tags into tokens. An example of this process applied to a Chinese sentence can be see in Figure 1.

Our implementation is based on the source code of LaserTagger.[6] We change the vocabulary optimization code to be able to handle Chinese sentences, in which words are not separated with spaces. We use character-level segmentation, as we find that the phrase vocabulary optimization algorithm achieves better results using character-level segmentation. This because segmentation errors are often present in the source sentences when using word-level segmentation, due to the sentences containing errors. Following the original paper, we use a phrase vocabulary of 500 phrases. A limitation of the sequence editing model is the small added phrase vocabulary, which consists only of the most frequently changed phrases between source and target sentence. Fortunately, our machine translation model is able to make up for this deficiency, as it is able to generate all of the words in the target vocabulary.

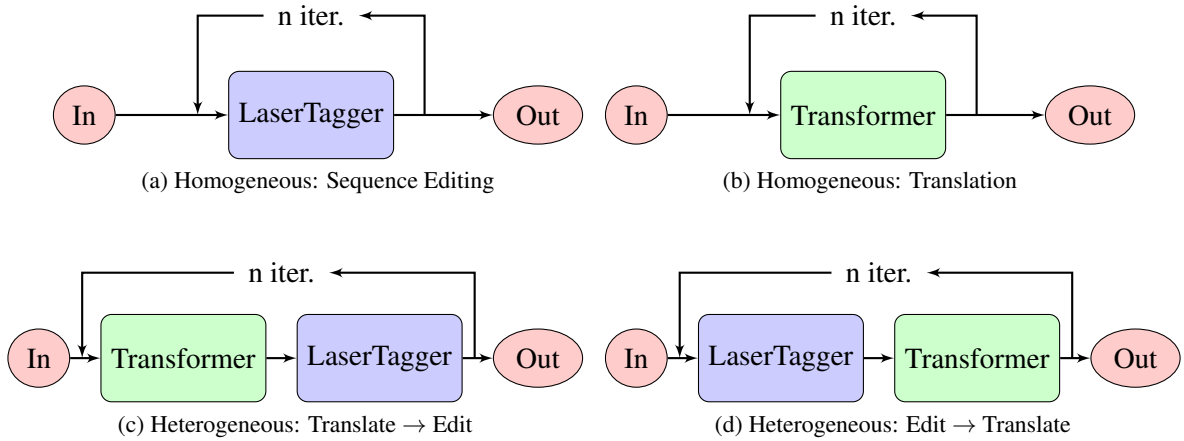
Training Settings When training our model, we use a batch size of 32 and train for three epochs. We use Adam (Kingma and Ba, 2014) as the optimizer with an initial learning rate of 3×10^{-5}. We perform a linear warm-up for 10% of the total training samples. Model checkpoints are saved every 1000 update steps. We use the model that performs best on our validation set as our final model.

3.4 Spell Checker

Spell checkers for Chinese work greatly differ from spell checkers for English because the sources of spelling errors in Chinese are entirely different from those in English. In general, spelling errors in Chinese are caused by **(1)** incorrectly selecting a character that looks similar to the correct character or **(2)** incorrectly selecting a character with similar pronunciation to the correct character.

Implementation We follow the steps outlined in the previous state-of-the-art work for Chinese GEC (Qiu and Qu, 2019) to create our spell-checker. The primary difference in our implementation is instead of a 5-gram language model, we use a Transformer (Vaswani et al., 2017) language model. This spelling checker works by iterating through a sentence one word at a time and checking if the word appears in a dictionary. If not, each character that makes up the word is replaced with each word in its confusion set to form a replacement candidate. After a list of candidate sentences are made, the Transformer language model picks the sentence with the highest probability.

[6] https://github.com/google-research/lasertagger



(a) Homogeneous: Sequence Editing

(b) Homogeneous: Translation

(c) Heterogeneous: Translate → Edit

(d) Heterogeneous: Edit → Translate

Figure 2: Configurations of recycle generation

LM Training We implement our language model using the Fairseq[7] toolkit (Ott et al., 2019). Optimization is performed using the Adam (Kingma and Ba, 2014) optimizer, with criterion set to cross entropy. We use beta values of 0.9 and 0.98 for Adam. We first set the learning rate to 10^{-7} and perform 4,000 warm-up updates. After the warm-up period, the learning rate is increased to 0.0005. Thereafter we use an inverse square root scheduler to decay the learning rate in proportion to the number of updates. Dropout between Transformer layers and attention layers is set to 0.1. We share weights between input and output embeddings. The batch size is set to a maximum of 2,048 tokens per batch, with one sample being a single sentence. The model is trained for a total of 50,000 updates, and achieves a perplexity of 66.47 on the validation set.

3.5 Recycle Generation

Recycle generation comes in two different varieties: homogeneous and heterogeneous. Homogeneous recycle generation is when a model performs several rounds of generation, where the output of the previous round of generation serves as input for the next round. Heterogeneous recycle generation integrates two or more different models that are trained to target different kinds of errors. Formally, homogeneous recycle generation can be described in the following way: given a family of models F, a trained model $f_\theta \in F$ with parameters θ and an input sequence $x \in X$,

$$f_\theta^n(x) = (f_\theta \circ \underset{n-2}{\cdots} \circ f_\theta)(x) = f_\theta(f_\theta(\underset{n-2}{\cdots}(x)))$$

where n is the number of iterations.

In a heterogeneous system, we would have $k \geq 2$ different families of models $F_1, \cdots, F_k$, and trained models $f_{1,\theta_1} \in F_1, \cdots, f_{k,\theta_k} \in F_k$ with parameters $\theta_1, \cdots \theta_k$ respectively. The composition order can be described with a sequence S of n model indices where $1 \leq j \leq k$ for each $j \in S$.

$$(f_{j_n,\theta_{j_n}} \circ \underset{n-2}{\cdots} \circ f_{j_1,\theta_{j_1}})(x) = f_{j_n,\theta_{j_n}}(f_{j_{n-1},\theta_{j_{n-1}}}(\underset{n-2}{\cdots}(x)))$$

In this work, both the *LaserTagger* sequence editing model and *Transformer* NMT model are used for recycle generation. We examine this system in regards to three factors: **(1)** homogeneous vs heterogeneous, **(2)** the number of iterations of generation, and **(3)** the *order* of composition. All configurations that we experiment with are summarized in Figure 2.

4 Experiments

The results of our recycle generation experiments are summarized in Table 4. The left column is homogeneous generation, the right column is heterogeneous generation, and each consecutive row is another iteration of generation.

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

Homogeneous generation For homogeneous recycle generation, $F_{0.5}$ score improves for three consecutive iterations, and decreases after 4 iterations for both our SE and NMT models. Precision decreases slightly with iteration count, but recall increases significantly, leading to an overall increase in F0.5 score. The sequence editing model LaserTagger performs better than our Transformer NMT model, achieving a highest F0.5 score of 32.27 on iteration 3, beating the best NMT score of 30.87 by a full 1.40 points.

Heterogeneous generation We find that applying our Transformer NMT model followed by our LaserTagger sequence editing model has the best performance. NMT followed by SE beats SE followed by NMT by 0.27 points. Furthermore, for SE followed by NMT the $F_{0.5}$ score consistently decreases after the second iteration. However, for NMT followed by SE, the $F_{0.5}$ score achieves a maximum on the fourth iteration, following a brief dip in performance on the third iteration.

Summary Overall, we can see that an increase in iteration count leads to a decrease in precision, but an increase in recall. Heterogeneous systems have much higher recall than homogeneous systems, with only slightly less precision, leading to much better overall performance. Finally, using our sequence editing model after our NMT model results in a higher F0.5 score.

Homogeneous Generation				**Heterogeneous Generation**			
	MaxMatch				**MaxMatch**		
Order	**P**	**R**	**F0.5**	**Order**	**P**	**R**	**F0.5**
NMT	37.54	17.38	30.47				
NMT^2	37.11	18.37	30.82	SE ∘ NMT	36.77	25.00	33.61
NMT^3	37.11	18.45	30.87	NMT ∘ SE ∘ NMT	36.14	25.88	33.49
NMT^4	37.09	18.45	30.85	$(SE \circ NMT)^2$	36.04	26.77	33.70
SE	39.80	17.21	31.53				
SE^2	38.13	19.34	31.93	NMT ∘ SE	36.87	24.10	33.34
SE^3	38.20	19.90	32.27	SE ∘ NMT ∘ SE	36.01	26.68	33.33
SE^4	37.98	19.88	32.13	$(NMT \circ SE)^2$	35.73	26.00	33.25

Table 4: Recycle generation experiments. NMT stands for neural machine translation system. SE stands for sequence editing system.

Comparison with state-of-the-art Following the results of our recycle generation experiments, we choose our best system $(SE \circ NMT)^2$ and apply our spell checker to the output of this system to serve as our final system output to compare with the previous state-of-the-art results. This comparison is presented in Table 5.

System	**MaxMatch**		
	P	**R**	**F0.5**
CS2S+Emb. (x4) (Ren et al., 2018)	**47.63**	12.56	30.57
Base+Emb. (i=2) + SC (Qiu and Qu, 2019)	36.88	18.94	31.01
*SC ∘ $(SE \circ NMT)^2$	36.79	**27.82**	**34.56**

Table 5: Comparison of our results to previous state-of-the-art results. * denotes our proposed system

The system denoted as CS2S+Emb. (x4) is an ensemble of 4 convolutional sequence to sequence models with pre-trained word embeddings. This system has very high precision, but suffers from low recall, resulting in the lowest overall $F_{0.5}$ score. The system denoted as Base+Emb. (i=2) + SC is the previous state of the art results on the NLPCC2018 dataset. It uses a sequence to sequence Transformer model with pre-trained word embeddings. Two iterations of recycle generation are applied, followed by

a spell checker. This system has much higher recall but lower precision than the previous approach. Our multi-model composition approach with recycle generation and spell checking beats the previous best results by over 3.5 $F_{0.5}$ score. Although our approach does not have the highest precision, it benefits from significantly improved recall in comparison to other approaches. This supports our hypothesis that our NMT model and sequence editing models correct different errors.

5 ERRANT for Chinese GEC

In the previous section, we saw that heterogeneous recycle generation was superior to homogeneous recycle generation. This was because heterogeneous recycle generation can significantly increase the recall of the GEC system, while only slightly sacrificing precision. This increase in recall leads us to believe that each of our models are attempting to correct different errors that are present in the sentences. Our precision not significantly decreasing, while simultaneously increasing recall means that our heterogeneous recycle generation system has vastly reduces false negatives while only slightly increasing false positives.

To verify our hypothesis that each model is more adept at correcting certain error types, we have to be able to calculate model performance specific to each of our four error types: redundant (R), missing (M), word selection (S), and word ordering (W). Unfortunately, the official Maxmatch (M2) scorer (Dahlmeier and Ng, 2012) does not have this capability. It can only provide the overall score of the system. In order to rectify this situation, we develop a scorer that can report the precision, recall, and $F_{0.5}$ score of our models with respect to our four error types. Another issue with the MaxMatch scorer worth mentioning is that it is known to *overestimate* model performance (Felice and Briscoe, 2015; Napoles et al., 2015).

The problem of error-type specific performance evaluation has already been solved for English GEC. The ERRANT scorer (Bryant et al., 2017) provides very detailed performance results for English GEC systems, by reporting the precision, recall, and $F_{0.5}$ scores with respect to over twenty-five different error categories.

The ERRANT scorer works in two main steps, **(1)** edit extraction (annotation) and **(2)** scoring. The edit extraction step works in the following way. First, the source and target sentences are tokenized and the part-of-speech for each token is computed. Next, the resulting tokenized sentences are aligned and labeled with *edit-level operations*: (M)atch, (D)elete, (I)nsert, and (S)ubstitute. After this, adjacent edit operations are merged using a rule-based approach. Finally, the resulting edits are classified into a specific error type, using the part of speech and edit-level operation information computed in steps (1) and (2). After annotation is performed, the system edits are compared with ground-truth edits, and the precision, recall, and $F_{0.5}$ score is computed.

5.1 Adapting ERRANT for Chinese Sentences

We adapt ERRANT to be able to annotate and score Chinese sentences. There are several challenges to do this. **(1)** Sentence alignment for Chinese is much different than English, because you cannot use Levenshtein distance as a heuristic. **(2)** In general, tokenizers for Chinese perform much worse than tokenizers for English. This is primarily because there are no spaces separating words in Chinese sentences, unlike English. In particular, we found that tokenization of the original (incorrect) sentence to give much worse results compared to the corrected sentence. This also leads to the part-of-speech tagger to often give incorrect results.

With these challenges in mind, we chose to implement our Chinese annotator in the following manner: we skip word-level tokenization and part-of-speech tagging in favor of character level tokenization. This means our scorer only gives results in terms of edit-level operations, which is the same as the original ground truth edits. We then adapt the original sentence alignment code from ERRANT [8] to be able to align Chinese sentences.

Edit Extraction The per-sentence edits (annotations) are extracted in three steps: tokenization, alignment, and merging. For tokenization, we use character-level tokenization. Alignment works by computing an alignment score for each pair of characters in the source and target sentence. Matching characters

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

give a score of zero. Insertion and deletion is each given a score of one. Substitution score is computed using a method similar to (Che et al., 2005), in which the Cilin (Mei et al., 1996) thesaurus is leveraged to give a similarity score between characters. Once the alignment score matrix is computed, the sequence with the lowest total score is returned. We use a simple merging strategy that merges consecutive sequences of edits of the same type.

5.2 Error-type Specific Performance

The results of re-scoring our systems with our scorer is presented in Table 6. We can see some clear differences in the strengths and weaknesses of our models. The NMT-based Transformer is best at correcting general substitution and word-ordering errors. The sequence editing LaserTagger model is better at correcting insertions and deletions. Finally, our spell checker is able to correct spelling errors (with are a special case of substitution errors) at high precision.

	NMT			**SE**			**SC**		
Error Type	**P**	**R**	**F0.5**	**P**	**R**	**F0.5**	**P**	**R**	**F0.5**
(R)emove	32.94	20.00	29.17	40.78	26.10	36.66	100.	0.0	0.0
(M)issing	29.49	14.85	24.64	34.23	17.00	28.46	100.	0.0	0.0
(S)ubstitute	32.03	11.54	23.64	34.53	7.79	20.47	80.85	4.12	17.1
(W)ord-order	49.18	27.78	42.61	56.76	18.75	40.38	100.	0.0	0.0

Table 6: System evaluations using the adapted ERRANT metric.

Comparison of SE and NMT Models The advantage that NMT models have over LaserTagger is that they have a much larger vocabulary, so they are not restricted to the amount of errors that they can correct. Any phrase that occurs in an error that is not in LaserTagger's small phrase vocabulary will not be able to be corrected by it. Currently, most word-order errors (W) are also not able to be corrected by LaserTagger. This is because LaserTagger lacks a method to re-arrange arbitrary subsequences of tokens inside of a sequence.

The main advantage LaserTagger has over NMT models is that it is very easy for LaserTagger to copy or delete a token from the source sentence to the output sentence. This is very useful in GEC, as errors are usually limited to just a few words in the source sentence, so most words can be copied directly to the target sentence (Zhao et al., 2019).

6 Conclusion

In this work, we propose a system for Chinese GEC that uses three different models: a NMT-based model, a sequence editing model, and a spell checker. We showed how these models can be composed using heterogeneous recycle generation into a system that achieves state of the art performance for Chinese GEC. Furthermore, we extended an automatic annotator and scorer for parallel error corpora to be able to handle Chinese sentences. We use this scorer to evaluate each models performance in terms of the four error types, and show how each model is adept at correcting different types of errors.

In the future, there is potential to combine the NMT-based model and the sequence editing model into a single model with both capabilities. Our first attempt at adapting ERRANT for Chinese GEC can also be improved upon by switching to word-level tokenization and adding POS tagging. This would allow for error classification to be performed, allowing each error to be classified into more distinct error types instead of just the four edit-level types that we currently use.

Acknowledgements

This research was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST-106-2923-E-002-012-MY3, MOST 108-2218-E-009-051, MOST 108-2634-F-002-017, and MOST 109-2634-F-002-034, and by Academia Sinica, Taiwan, under grant AS-TP-107-M05.

References

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (wmt19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy, August. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada, July. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy, August. Association for Computational Linguistics.

Wanxiang Che, Jianmin Jiang, Zhong Su, Yue Pan, and Ting Liu. 2005. Improved-edit-distance kernel for chinese relation extraction. In *Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada, June. Association for Computational Linguistics.

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing. *arXiv preprint arXiv:1906.08104*.

Mariano Felice and Ted Briscoe. 2015. Towards a standard evaluation method for grammatical error detection and correction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 578–587.

Kai Fu, Jin Huang, and Yitao Duan. 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*, pages 341–350. 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, Volume 2 (Short Papers)*, pages 284–290, New Orleans, Louisiana, June. Association for Computational Linguistics.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, Florence, Italy, August. Association for Computational Linguistics.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11179–11189.

Hen-Hsen Huang, Yen-Chi Shao, and Hsin-Hsi Chen. 2016. Chinese preposition selection for grammatical error diagnosis. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 888–899.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242, Hong Kong, China, November. Association for Computational Linguistics.

Jared Lichtarge, Christopher Alberti, Shankar Kumar, Noam Shazeer, and Niki Parmar. 2018. Weakly supervised grammatical error correction using iterative decoding. *ArXiv*, abs/1811.01710.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *EMNLP-IJCNLP*.

Jiaju Mei, Yiming Lan, Yunqi Gao, and Hongxiang Yin. 1996. Chinese thesaurus tongyici cilin (2nd edtion).

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June. Association for Computational Linguistics.

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

Z. Qiu and Y. Qu. 2019. A two-stage model for chinese grammatical error correction. *IEEE Access*, 7:146772–146777.

Hongkai Ren, Liner Yang, and Endong Xun. 2018. A sequence to sequence learning for chinese grammatical error correction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 401–410. Springer.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 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)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at sighan bake-off 2013. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 35–42.

Yuanyuan Zhao, Nan Jiang, Weiwei Sun, and Xiaojun Wan. 2018. Overview of the nlpcc 2018 shared task: grammatical error correction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 439–445. Springer.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota, June. Association for Computational Linguistics.