A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction

Shamil Chollampatt,1 Hwee Tou Ng1,2
1NUS Graduate School for Integrative Sciences and Engineering
2Department of Computer Science
National University of Singapore
shamil@u.nus.edu, nght@comp.nus.edu.sg

Abstract
We improve automatic correction of grammatical, orthographic, and collocation errors in text using a multilayer convolutional encoder-decoder neural network. The network is initialized with embeddings that make use of character N-gram information to better suit this task. When evaluated on common benchmark test data sets (CoNLL-2014 and JF-LEG), our model substantially outperforms all prior neural approaches on this task as well as strong statistical machine translation-based systems with neural and task-specific features trained on the same data. Our analysis shows the superiority of convolutional neural networks over recurrent neural networks such as long short-term memory (LSTM) networks in capturing the local context via attention, and thereby improving the coverage in correcting grammatical errors. By ensembling multiple models, and incorporating an N-gram language model and edit features via rescoring, our novel method becomes the first neural approach to outperform the current state-of-the-art statistical machine translation-based approach, both in terms of grammaticality and fluency.

Introduction
With the increasing number of non-native learners and writers of the English language around the globe, the necessity to improve authoring tools such as error correction systems is increasing. Grammatical error correction (GEC) is a well-established natural language processing (NLP) task that deals with building systems for automatically correcting errors in written text, particularly in non-native written text. The errors that a GEC system attempts to correct are not limited to grammatical errors, but also include spelling and collocation errors.

GEC in English has gained much attention within the NLP community recently. The phrase-based statistical machine translation (SMT) approach has emerged as the state-of-the-art approach for this task (Chollampatt and Ng 2017; Junczys-Dowmunt and Grundkiewicz 2016), in which GEC is treated as a translation task from the language of “bad” English to the language of “good” English. The translation model is learned using parallel error-corrected corpora (source text that contains errors and their corresponding corrected target text). Although neural network (NN) models have been used as features to improve the generalization of the SMT approach (Chollampatt, Taghipour, and Ng 2016), SMT still suffers from limitations in accessing the global source and target context effectively. The treatment of words and phrases as discrete entities during decoding also limits its generalization capabilities. To this end, several neural encoder-decoder approaches were proposed for this task (Xie et al. 2016; Yuan and Briscoe 2016; Ji et al. 2017; Schmaltz et al. 2017). However, their performance still falls substantially behind state-of-the-art SMT approaches.

All prior neural approaches for GEC relied on using recurrent neural networks (RNNs). In contrast to previous neural approaches, our neural approach to GEC is based on a fully convolutional encoder-decoder architecture with multiple layers of convolutions and attention (Gehring et al. 2017). Our analysis shows that convolutional neural networks (CNNs) can capture local context more effectively than RNNs as the convolution operations are performed over smaller windows of word sequences. Most grammatical errors are often localized and dependent only on the nearby words. Wider contexts and interaction between distant words can also be captured by a multilayer hierarchical structure of convolutions and an attention mechanism that weights the source words based on their relevance in predicting the target word. Moreover, only a fixed number of non-linearities are performed on the input irrespective of the input length whereas in RNNs, the number of non-linearities is proportional to the length of the input, diminishing the effects of distant words.

We further improve the performance by ensembling multiple models. Contrary to prior neural approaches, we use a simpler pre-processing method to alleviate the unknown word problem (Sennrich, Haddow, and Birch 2016). Rare words are split into multiple frequent sub-words using a byte pair encoding (BPE) algorithm. One of the major weaknesses of prior neural approaches is that they do not incorporate task-specific features nor utilize large native English corpora to good effect. We use such English corpora in our encoder-decoder model to pre-train the word vectors to be used for initializing the embeddings in the encoder and decoder. We also train an N-gram language model to be used as a feature along with edit operation count features in rescoring to produce an overall better output.

To summarize, this paper makes the following contribu-
tions: (1) We successfully employ a convolutional encoder-decoder model trained on BPE tokens as our primary model to achieve state-of-the-art performance for GEC. Ours is the first work to use fully convolutional neural networks for end-to-end GEC. (2) We exploit larger English corpora to pre-train word embeddings and to train an N-gram language model to be used as a feature in a rescorer that is trained to optimize the target metric using minimum error rate training (Och 2003). (3) We conduct a comparison of attention mechanisms in typical recurrent architectures and our models, and perform error type performance analysis to identify the strengths of our approach over the current state-of-the-art SMT approach.

**Related Work**

GEC gained much attention within the NLP community after the CoNLL-2014 shared task (Ng et al. 2014) was organized. The shared task dealt with the correction of all grammatical errors in English essays. Since then, the test set for the shared task has been used to benchmark GEC systems. Statistical machine translation has emerged as the state-of-the-art approach (Chollampatt and Ng 2017) due to its ability to correct various types of errors and complex error patterns, whereas previous approaches relied on building error type-specific classifiers (Dahlmeier, Ng, and Ng 2012; Rozovskaya et al. 2014). The SMT framework largely benefits from its ability to incorporate large error-corrected parallel corpora like the publicly available Lang-8 corpus (Mizumoto et al. 2011), additional English corpora for training robust language models (LMs), task-specific features (Junczys-Dowmunt and Grundkiewicz 2016), and neural models (Chollampatt, Taghipour, and Ng 2016). However, SMT-based systems suffer from limited generalization capabilities compared to neural approaches and are unable to access longer source and target contexts effectively. To address these issues, several neural encoder-decoder approaches relying on RNNs were proposed for GEC.

**Neural Encoder-Decoder Approaches to GEC**

Yuan and Briscoe (2016) first applied a popular neural machine translation model, RNNSearch (Bahdanau, Cho, and Bengio 2015), consisting of a bidirectional RNN encoder and an attention-based RNN decoder. They additionally made use of an unsupervised word alignment model and a word-level statistical translation model to replace unknown words in the output. However, they trained their systems on 1.9M sentence pairs from the professionally annotated, non-public Cambridge Learner Corpus (CLC), making their models hard to replicate and compare with.

Xie et al. (2016) proposed the use of a character-level recurrent encoder-decoder network for GEC. They trained their models on the publicly available NUCLE (Dahlmeier, Ng, and Wu 2013) and Lang-8 corpora, along with synthesized examples for frequent error types. They also incorporated an N-gram LM trained on a small subset of the Common Crawl corpus (2.2B N-grams) during decoding to achieve an F0.5 score of 39.97 on the CoNLL-2014 test set. They further used a supervised edit classifier trained on character and word-level edit operation and pre-trained word embedding features to remove spurious edits and improve the F0.5 score to 40.56.

Ji et al. (2017) proposed a hybrid word-character model based on the hybrid machine translation model of (Luong and Manning 2016), by adding nested levels of attention at the word and character level. Similar to (Yuan and Briscoe 2016), they also made use of the non-public CLC corpus in training in addition to Lang-8 and NUCLE, resulting in 2.6M sentence pairs. By further adding a web-scale Common Crawl LM that was used in (Junczys-Dowmunt and Grundkiewicz 2016) in a rescoring step, they achieved an F0.5 score of 45.15 on the CoNLL-2014 test set. Their rescorer was trained using a simple grid search with fixed step size to get the feature weights and did not make use of task-specific features, whereas we use minimum error rate training (Och 2003) to find optimal feature weights and use edit operation features and LM features.

More recently, Schmaltz et al. (2017) used a word-level bidirectional LSTM network trained on Lang-8 and NUCLE (1.4M sentence pairs) with edit operations (insertions, deletions, and substitutions) marked with special tags in the target sentences. Their untuned model and the baseline that did not have edit operation tags marked yielded a high precision and a low recall. However, when they tuned the weights for the edit operations using a grid search maximizing F0.5, their recall went up. Without using any additional models or corpora, their approach achieved F0.5 score of 41.37 on the CoNLL-2014 test set. Their edit operation tagging method and tuning also implicitly modeled edit operation weights. We model edit operations explicitly in our approach by counting and using them as weighted features in our rescorer.

**A Multilayer Convolutional Encoder-Decoder Neural Network**

Encoder-decoder models are most widely used for machine translation from a source language to a target language. Similarly, an encoder-decoder model can be employed for GEC, where the encoder network is used to encode the potentially erroneous source sentence in vector space and a decoder network generates the corrected output sentence by using the source encoding. The attention mechanism (Bahdanau, Cho, and Bengio 2015) selectively weights different parts of the source sentence during decoding, allowing for a different encoding of the source sentence at every decoding time step. We build our models based on an encoder-decoder architecture with multiple layers of convolutions and attention mechanisms, similar to its use in MT by (Gehring et al. 2017). The models are trained on words with rare words segmented into sub-words (Sennrich, Haddow, and Birch 2016).

**Model**

Consider an input source sentence $S$ given as a sequence of $m$ source tokens $s_1, \ldots, s_m$ and $s_i \in V_s$, where $V_s$ is the source vocabulary. The last source token, $s_m$, is a special end-of-sentence marker token. The source tokens are embedded in continuous space as $\mathbf{s}_1, \ldots, \mathbf{s}_m$. The embedding
Each output vector of the final encoder layer, \( h^{l}_{i} \in \mathbb{R}^{h} \), is linearly mapped to get the encoder output vector, \( e_{i} \in \mathbb{R}^{d} \), using weights \( W_{e} \in \mathbb{R}^{d \times h} \) and biases \( b_{e} \in \mathbb{R}^{d} \):

\[
e_{i} = W_{e} h^{l}_{i} + b_{e} \quad i = 1, \ldots, m
\]

Now, consider the generation of the target word \( t_{n} \) at the \( n^{th} \) time step in decoding, with \( n - 1 \) target words previously generated. For the decoder, paddings are added at the beginning. The two paddings, beginning-of-sentence marker and the previously generated tokens, are embedded as \( t_{-2}, t_{-1}, t_{0}, t_{1}, \ldots, t_{n-1} \) in the same way as source token embeddings are computed. Each embedding \( t_{j} \in \mathbb{R}^{d} \) is linearly mapped to \( g^{j}_{0} \in \mathbb{R}^{h} \) and passed as input to the first decoder layer. In each decoder layer, convolution operations followed by non-linearities are performed on the previous decoder layer’s output vectors \( g^{j-1}_{n} \), where \( j = 1, \ldots, n \):

\[
y_{j} = \text{GLU}(\text{Conv}(g^{j-1}_{j-3} \circ g^{j-1}_{j-2} \circ g^{j-1}_{j-1}))
\]

where \( \text{Conv}(\cdot) \) and \( \text{GLU}(\cdot) \) represent convolutions and non-linearities respectively, and \( y_{j} \) becomes the decoder state at the \( j^{th} \) time step in the \( j^{th} \) decoder layer. The number and size of convolution filters are the same as those in the encoder.

Each decoder layer has its own attention module. To compute attention at layer \( l \) before predicting the target token at the \( n^{th} \) time step, the decoder state \( y_{n}^{l} \in \mathbb{R}^{h} \) is linearly mapped to a \( d \)-dimensional vector with weights \( W_{d} \in \mathbb{R}^{d \times h} \) and biases \( b_{d} \in \mathbb{R}^{d} \), adding the previous target token’s embedding:

\[
z_{n}^{l} = W_{d} y_{n}^{l} + b_{d} + t_{n-1}
\]

The attention weights \( a_{n,i}^{l} \) are computed by a dot product of the encoder output vectors \( e_{1}, \ldots, e_{m} \) with \( z_{n}^{l} \) and normalized by a softmax:

\[
a_{n,i}^{l} = \frac{\exp(e_{i}^{\top} z_{n}^{l})}{\sum_{k=1}^{m} \exp(e_{k}^{\top} z_{n}^{l})} \quad i = 1, \ldots, m
\]

The source context vector \( x_{n}^{l} \) is computed by applying the attention weights to the summation of the encoder output vectors and the source embeddings. The addition of the source embeddings helps to better retain information about the source tokens:

\[
x_{n}^{l} = \sum_{i=1}^{m} a_{n,i}^{l} (e_{i} + s_{i})
\]

The context vector \( x_{n}^{l} \) is then linearly mapped to \( c_{n}^{l} \in \mathbb{R}^{h} \). The output vector of the \( l^{th} \) decoder layer, \( g_{n}^{l} \), is the summation of \( c_{n}^{l} \), \( y_{n}^{l} \), and the previous layer’s output vector \( g_{n}^{l-1} \):

\[
g_{n}^{l} = y_{n}^{l} + c_{n}^{l} + g_{n}^{l-1}
\]

The final decoder layer output vector \( g_{n}^{5757} \) is linearly mapped to \( d_{n} \in \mathbb{R}^{d} \). Dropout (Srivastava et al. 2014) is applied at the decoder outputs, embeddings, and before every encoder and decoder layer. The decoder output vector is then mapped to the target vocabulary size (|\( V_{t} \)|) and softmax is computed to obtain target word probabilities:

\[
o_{n} = W_{o} d_{n} + b_{o} \quad W_{o} \in \mathbb{R}^{|V_{t}| \times d}, b_{o} \in \mathbb{R}^{|V_{t}|}
\]
\[ p(t_n = w_i | t_1, \ldots, t_{n-1}, S) = \frac{\exp(o_{n,i})}{\sum_{k=1}^{l} \exp(o_{n,k})} \]

where \( w_i \) is the \( i \)th word in the target vocabulary \( V_t \).

### Pre-Training of Word Embeddings

We initialize the word embeddings for the source and target words with pre-trained word embeddings learned from a large English corpus. Rare words in this English corpus are split into BPE-based sub-word units as we use similar preprocessing for the parallel corpus that is used to train the network. The word embeddings are computed by representing a word as a bag of character \( N \)-grams and summing the skip-gram embeddings of these character \( n \)-gram sequences, using the fastText tool (Bojanowski et al. 2017). These embeddings have information about the underlying morphology of words and was empirically found to perform better than initializing the network randomly or using word2vec (Mikolov et al. 2013) embeddings, which treat words as separate entities and have no information about the character sequences that make up the words.

#### Training

The model is trained using the negative log-likelihood loss function:

\[
L = -\frac{1}{N} \sum_{i=1}^{N} \frac{1}{T_i} \sum_{j=1}^{T_i} \log(p(t_{i,j} | t_{i,1}, \ldots, t_{i,j-1}, S))
\]

where \( N \) is the number of training instances in a batch, \( T_i \) is the number of tokens in the \( i \)th reference sentence, \( t_{i,j} \) is the \( j \)th target word in the reference correction for the \( i \)th training instance. The parameters are optimized using Nesterov’s Accelerated Gradient Descent (NAG) with a simplified formulation for Nesterov’s momentum (Bengio, Boulanger-Lewandowski, and Pascanu 2013).

#### Decoding

The encoder-decoder model estimates the probability of target words given the erroneous source sentence \( S \). The best sequence of target words is obtained by a left-to-right beam search. In a beam search, the top \( b \) probable candidates at every decoding step is retained. The top-scoring candidate in the beam at the end of the search will be the correction hypothesis. The model score of a hypothesis is the sum of the log probabilities of the hypothesis words computed by the network. We also perform ensembling during decoding by averaging the predictions from multiple models in order to compute the log probability scores of the hypothesis words. The models used for ensembling have the same architecture but are trained with different random initializations.

#### Rescoring

In order to incorporate task-specific features and large language models, we re-score the final beam candidates using a log-linear framework. The score of a correction hypothesis sentence \( T \) given the source sentence \( S \) is given by:

\[
\text{score}(T, S) = \sum_{i=1}^{F} \lambda_i f_i(T, S)
\]

where, \( \lambda_i \) and \( f_i \) are the \( i \)th feature weight and feature function respectively, and \( F \) is the number of features. The feature weights are computed by minimum error rate training (MERT) (Och 2003) on the development set. We use the following sets of features in rescoring in addition to the model score of the hypothesis:

1. **Edit operation (EO)** features: Three features denoting the number of token-level substitutions, deletions, and insertions between the source sentence and the hypothesis sentence.
2. **Language model (LM)** features: Two features, a 5-gram language model score (i.e., the sum of log probabilities of 5-grams in the hypothesis sentence) and the number of words in the hypothesis. Similar to state-of-the-art methods, the language model is trained on the web-scale Common Crawl corpus (Chollampatt and Ng 2017; Junczys-Dowmunt and Grundkiewicz 2016).

### Experimental Setup

#### Datasets

We use two public datasets from prior work, Lang-8 (Mizumoto et al. 2011) and NUCLE (Dahlmeier, Ng, and Wu 2013), to create our parallel data. Along with the sentence pairs from NUCLE, we extract and use the English sentence pairs in Lang-8 by selecting essays written by English learners and removing non-English sentences from them using a language identification tool\(^1\). Sentence pairs that are unchanged on the target side are discarded from the training set. A subset of this data, 5.4K sentence pairs from NUCLE, is taken out to be used as the development data for model selection and training the rescorer. The remaining parallel data that is used for training the encoder-decoder NN consists of 1.3M sentence pairs (18.05M source words and 21.53M target words). We also make use of the larger English corpora from Wikipedia (1.78B words) for pre-training the word embeddings, and a subset of the Common Crawl corpus (94B words) for training the language model for rescoring. Corpora of similar size from the Common Crawl have been used by leading GEC systems (Chollampatt and Ng 2017; Ji et al. 2017; Junczys-Dowmunt and Grundkiewicz 2016).

#### Evaluation

Our evaluation setting is the same as that in the CoNLL-2014 shared task. We evaluate our models and compare them to previous systems on the official CoNLL-2014 test set using the F\(_{0.5}\) score computed using the MaxMatch scorer (Dahlmeier and Ng 2012). Following prior work, we analyze our neural model choices and perform ablation studies on the CoNLL-2013 shared task test set.

We also evaluate the fluency of our model outputs on the recently released JFLEG development and test sets (Napoles, Sakaguchi, and Tetreault 2017), which have fluency-based rewrites of learner-written sentences done by native writers in order to make the sentences native-sounding and error-free. The GLEU metric is used to as-

\(^1\)https://github.com/saffsd/langid.py
Table 1: Results on the CoNLL-2014 test set. For single models (MLConv and MLConvembed), average precision, recall, and F0.5 of 4 models (trained with different random initializations) are reported. (4 ens.) refers to the ensemble decoding of these 4 models. +EO and +LM refer to re-scoring using edit operation and language model features, respectively. +SpellCheck denotes the addition of the publicly available spell checker proposed in (Chollampatt and Ng 2017). L8 refers to the Lang-8 corpus, CC refers to Common Crawl, CLC refers to the non-public Cambridge Learner Corpus, and SP refers to the corpus of misspellings. A smaller subset of CC (2.2B words) was used in (Xie et al. 2016) compared to the rest (94B – 97B words).

| System                         | Parallel Data | Is Data Public? | Other Corpora | CoNLL-2014 Test Set |
|-------------------------------|---------------|-----------------|---------------|---------------------|
| Baselines                     |               |                 |               |                     |
| SMT                           | L8, NUCLE     | Yes             | –             | 57.94               |
| SMT +NNJM                     | L8, NUCLE     | Yes             | –             | 58.38               |
| This work (no additional corpora) |               |                 |               |                     |
| MLConv                        | L8, NUCLE     | Yes             | –             | 59.68               |
| MLConv (4 ens.)               | L8, NUCLE     | Yes             | –             | 67.06               |
| MLConv (4 ens.) +EO           | L8, NUCLE     | Yes             | –             | 62.36               |
| This work (using additional corpora) |               |                 |               |                     |
| MLConvembed                   | L8, NUCLE     | Yes             | Wiki          | 60.90               |
| MLConvembed (4 ens.)          | L8, NUCLE     | Yes             | Wiki          | 68.13               |
| MLConvembed (4 ens.) +EO      | L8, NUCLE     | Yes             | Wiki          | 63.12               |
| MLConvembed (4 ens.) +EO +LM  | L8, NUCLE     | Yes             | Wiki, CC      | 65.18               |
| MLConvembed (4 ens.) +EO +LM +SpellCheck | L8, NUCLE | Yes             | Wiki, CC, SP  | 65.49               |
| Prior encoder-decoder approaches |               |                 |               |                     |
| (Ji et al. 2017) with LM      | L8, NUCLE, CLC| No              | CC            | 45.15               |
| (Ji et al. 2017) without LM   | L8, NUCLE, CLC| No              | –             | 41.53               |
| (Schmalzt et al. 2017)        | L8, NUCLE     | Yes             | –             | 41.37               |
| (Xie et al. 2016) with LM, Edit Classifier | L8, NUCLE | Yes             | CC (small)    | 40.56               |
| (Yuan and Briscoe 2016)       | CLC           | No              | –             | 39.90               |
| State-of-the-art systems      |               |                 |               |                     |
| (Chollampatt and Ng 2017) +SpellCheck | L8, NUCLE | Yes             | Wiki, CC, SP  | 53.14               |
| (Chollampatt and Ng 2017)     | L8, NUCLE     | Yes             | Wiki, CC      | 51.70               |
| (Junczys-Dowmunt and Grundkiewicz 2016) | L8, NUCLE | Yes             | Wiki, CC      | 49.49               |

Sess fluency of corrected text when the error-span and error-type annotations are not provided (Napoles et al. 2015). We also calculate the F0.5 score after automatically extracting the annotation span using the scripts released with the JF-LEG dataset.

Model and Training Details

We extend the publicly available PyTorch-based implementation\(^2\) for training multilayer convolutional models initialized with pre-trained embeddings. Both source and target embeddings are of 500 dimensions. Each of the source and target vocabularies consists of 30K most frequent BPE tokens from the source and target side of the parallel data, respectively. Pre-training is done using fastText with one pass on the Wikipedia corpus using a skip-gram model with a window size of 5. Character N-gram sequences of size between 3 and 6 (both inclusive) are used to compute the word embeddings and other parameters are kept to their default values. The embeddings are updated during training of the encoder-decoder NN. Each of the encoder and decoder is made up of seven convolutional layers, with a convolution window width of 3. The number of layers in the encoder and decoder is set based on development set performance after experimenting with 5, 7, and 9 layers. Output of each encoder and decoder layer is of 1024 dimensions. Dropout with probability 0.2 is applied on the embeddings, convolution layers, and decoder output. We train every model simultaneously on 3 NVIDIA Titan X GPUs with a batch size of 32 on each GPU and perform validation after every epoch concurrently on another NVIDIA Titan X GPU. A learning rate of 0.25 is used with a learning rate annealing factor of 0.1 and a momentum value of 0.99. We use early stopping and select the best model based on the F0.5 score on the development set. Training a single model takes around 18 hours. During decoding, a beam width of 12 is used.

Baselines

We compare our systems to all prior neural approaches for GEC and two state-of-the-art (SOTA) systems. The first SOTA system (Junczys-Dowmunt and Grundkiewicz 2016) employs a word-level SMT approach with task-specific features and a web-scale LM trained on the Common Crawl corpus. The second SOTA system (Chollampatt and Ng 2017) adds an adapted neural network joint model (NNJM) to a word-level SMT system with task-specific features and a web-scale LM, with further improvement by spelling error correction using a character-level SMT system. In order to compare our neural approach to the SMT approach without using other English corpora, we create two baselines using released models of the SOTA system (Chollampatt and Ng 2017). The first (SMT +NNJM in Table 1) is this word-level SMT-based system retuned after removing

---

\(^2\)https://github.com/facebookresearch/fairseq-py
all subsidiary models that make use of additional English corpora such as the word-class LM and the web-scale Common Crawl LM. This system has an adapted NNJM and an operation sequence model (OSM), both trained on the parallel data, and has a single LM trained on the target side of the parallel data. Another non-neural SMT baseline (SMT in Table 1) is created by further removing the adapted NNJM and retuning on our development set.

Experiments and Results

Evaluation on Benchmark Corpora

We evaluate our systems based on the grammaticality and fluency of their output sentences.

Grammaticality We first evaluate different variants of our system on the CoNLL-2014 test data (Table 1). Our single model without using any additional corpora or rescoring (MLConv) achieves 45.36 F0.5. After ensembling four models (4 ens.), the performance reaches 48.05 F0.5 and outperforms the previous best neural model without LM (Ji et al. 2017) (41.53 F0.5) by a large margin of 6.52 F0.5, despite the latter using much more training data including the non-public CLC. Our neural systems also substantially outperform the two comparable SMT baselines, ‘SMT’ and ‘SMT +NNJM’. When re-scoring is performed with edit operation (+EO) features, the performance goes up to 49.78 F0.5, outperforming a strong SMT-based system (Junczys-Dowmunt and Grundkiewicz 2016) that uses task-specific features and a web-scale Common Crawl language model. Our system, on the other hand, achieves this level of performance without using any additional English corpora or pre-trained word embeddings. When we train our models by initializing with pre-trained fastText word embeddings (MLConvembed), decode using an ensemble of four models, and rescore with edit operation features, the performance reaches 50.70 F0.5.

After adding the web-scale LM in rescoring (+LM), our approach reaches 54.13 F0.5, outperforming the best previous published result of (Chollampatt and Ng 2017) (F0.5 = 53.14) that additionally uses a spelling correction component trained on a spelling corpus. This improvement is statistically significant (p < 0.001). When we make use of the spelling correction component in (Chollampatt and Ng 2017) (+SpellCheck), our performance reaches 54.79, a statistically significant improvement of 1.65 F0.5 (p < 0.001) over the best previous published result, and establishes the new state of the art for English GEC. All statistical significance tests were performed using sign test with bootstrap resampling on 100 samples.

Fluency We also measure the fluency of the outputs on the JFLEG development and test sets (Table 2). Our system with rescoring using edit operation features outperforms the state-of-the-art system with a web-scale LM without spell checking (Chollampatt and Ng 2017) on both datasets and metrics. This is without adding the web-scale LM to our system. After adding the web-scale LM and using the spell checker, our method achieves the best reported GLEU and F0.5 scores on these datasets. It is worth noting that our models achieve this level of performance without tuning on the JFLEG development set.

Encoder and Decoder Architecture

We analyze the performance of various network architectures without using pre-trained word embeddings on the CoNLL-2013 test set (Table 3). We experiment with using a bidirectional LSTM in the encoder and an attentional LSTM decoder with a soft attention mechanism (Bahdanau, Cho, and Bengio 2015) (BiLSTM in Table 3), and compare it to single layer convolutional (SLConv) as well as our proposed multilayer convolutional (MLConv) encoder and decoder models. BiLSTM can capture the entire sentence context from left and right for each input word, whereas SLConv captures only a few surrounding words (equal to the filter width of 3). However, MLConv captures a larger surrounding context (7 layers × filter width 3 = 21 tokens) more effectively, causing it to outperform both SLConv and BiLSTM.

It is interesting to note that the BiLSTM model has a higher precision than the MLConv model, although its recall is lower. We analyze the attention weights of both models (Figure 2) on an example sentence from the CoNLL-2013 test set. The attention weights shown for the MLConv model is the averaged attention weights of all decoder layers. It can be seen that BiLSTM produces a sharper distribution placing higher weights on matching source words as opposed to MLConv which places noticeable probability mass on the surrounding context words also. We observed this trend for all other examples that we tried. This could be the reason that causes BiLSTM to frequently output the source words, leading to a fewer number of proposed corrections and consequently, a higher precision. This analysis demonstrates the ability of MLConv in capturing the context better, thereby favoring more corrections than copying of the source words.
Figure 2: Visualization of attention weights of BiLSTM and MLConv models. y-axis shows the target words and x-axis shows the source words.

Table 4: Results of different embedding initializations on the CoNLL-2013 test set.

| Initialization | Prec.  | Recall | F0.5  |
|---------------|--------|--------|-------|
| Random        | 51.90  | 12.59  | 31.96 |
| Word2vec      | 52.80  | 12.80  | 32.49 |
| fastText      | 51.08  | 13.63  | 32.97 |

Initialization with Pre-trained Embeddings

We assess various methods of initializing the source and target word embeddings. Table 4 shows the results of initializing the embeddings randomly as well as with word2vec and fastText on the CoNLL-2013 test set. We train skip-gram models with word2vec and use parameters identical to those we use for fastText. fastText embeddings have access to the character sequences that make up the words and hence are better suited to learn word representations taking morphology into account. We also find that initializing with fastText works well empirically, and hence we choose these embeddings to initialize our network when evaluating on benchmark test datasets.

Analysis and Discussion

We perform error type-specific performance comparison of our system and the state-of-the-art (SOTA) system (Chollampatt and Ng 2017), using the recently released ERRANT toolkit (Bryant, Felice, and Briscoe 2017) on the CoNLL-2014 test data based on F0.5. ERRANT relies on a rule-based framework to identify the error type of corrections proposed by a GEC system. The results on four common error types are shown in Figure 3. We find that our ensembled model with the rescorer (+EO+LM) performs competitively on preposition errors, and outperforms the SOTA system on noun-number, determiner, and subject-verb agreement errors. One of the weaknesses of SMT-based systems is in correction of subject-verb agreement errors, because a verb and its subject can be very far apart within a source sentence. On the other hand, even our single model (MLConv_embed) without rescoring is superior to the SOTA SMT-based system in terms of subject-verb agreement errors, since it has access to the entire source context through the global attention mechanism and to longer target context through multiple layers of convolutions in the decoder.

From our analysis, we find that a convolutional encoder-decoder NN captures the context more effectively compared to an RNN and achieves superior results. However, RNNs can give higher precision, so a combination of both approaches could be investigated in future. Improved language modeling has been previously shown to improve GEC performance considerably. We leave it to future work to explore the integration of web-scale LM during beam search and the fusion of neural LMs into the network. We also find that a simple preprocessing method that segments rare words into sub-words effectively deals with the rare word problem for GEC, and performs better than character-level models and complex word-character models.

Conclusion

We use a multilayer convolutional encoder-decoder neural network for the task of grammatical error correction and achieve significant improvements in performance compared to all previous encoder-decoder neural network approaches. We utilize large English corpora to pre-train and initialize the word embeddings and to train a language model to rescore the candidate corrections. We also make use of edit operation features during rescoring. By ensembling multiple neural models and rescoring, our novel method achieves improved performance on both CoNLL-2014 and JFLEG data sets, significantly outperforming the current leading SMT-based systems. We have thus fully closed the large performance gap that previously existed between neural and statistical approaches for this task. The source code and model files used in this paper are available at https://github.com/nusnlp/mlconvgec2018.
Acknowledgements
We thank the anonymous reviewers for their feedback. This research was supported by Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2013-T2-1-150.

References
Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In Proc. ICLR.
Bengio, Y.; Boulanger-Lewandowski, N.; and Pascanu, R. 2013. Advances in optimizing recurrent networks. In Proc. IJCASS.
Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. TACL 5:135–146.
Bryant, C.; Felice, M.; and Briscoe, T. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In Proc. ACL.
Chollampatt, S., and Ng, H. T. 2017. Connecting the dots: Towards human-level grammatical error correction. In Proc. of the 12th Workshop on Innovative Use of NLP for Building Educational Applications.
Chollampatt, S.; Taghipour, K.; and Ng, H. T. 2016. Neural network translation models for grammatical error correction. In Proc. IJCAI.
Dahlmeier, D., and Ng, H. T. 2012. Better evaluation for grammatical error correction. In Proc. NAACL.
Dahlmeier, D.; Ng, H. T.; and Ng, E. J. F. 2012. NUS at the HOO 2012 shared task. In Proc. of the 7th Workshop on the Innovative Use of NLP for Building Educational Applications.
Dahlmeier, D.; Ng, H. T.; and Wu, S. M. 2013. Building a large annotated corpus of learner English: The NUS Corpus of Learner English. In Proc. of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications.
Dauphin, Y. N.; Fan, A.; Auli, M.; and Grangier, D. 2016. Language modeling with gated convolutional networks. arXiv preprint arXiv:1612.08083.
Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. In Proc. ICML.
Ji, J.; Wang, Q.; Toutanova, K.; Gong, Y.; Truong, S.; and Gao, J. 2017. A nested attention neural hybrid model for grammatical error correction. In Proc. ACL.
Junczys-Dowmunt, M., and Grundkiewicz, R. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In Proc. EMNLP.
Luong, M.-T., and Manning, C. D. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In Proc. ACL.
Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In Proc. NIPS.
Mizumoto, T.; Komachi, M.; Nagata, M.; and Matsumoto, Y. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In Proc. IJCNLP.
Napoles, C.; Sakaguchi, K.; Post, M.; and Tetreault, J. 2015. Ground truth for grammatical error correction metrics. In Proc. ACL/IJCNLP.
Napoles, C.; Sakaguchi, K.; and Tetreault, J. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In Proc. EACL.
Ng, H. T.; Wu, S. M.; Briscoe, T.; Hadiwinoto, C.; Susanto, R. H.; and Bryant, C. 2014. The CoNLL-2014 shared task on grammatical error correction. In Proc. CoNLL: Shared Task.
Och, F. J. 2003. Minimum error rate training in statistical machine translation. In Proc. ACL.
Rozovskaya, A.; Chang, K.-W.; Sammons, M.; Roth, D.; and Habash, N. 2014. The Illinois-Columbia system in the CoNLL-2014 shared task. In Proc. CoNLL: Shared Task.
Schmaltz, A.; Kim, Y.; Rush, A. M.; and Shieber, S. 2017. Adapting sequence models for sentence correction. In Proc. EMNLP.
Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural machine translation of rare words with subword units. In Proc. ACL.
Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. JMLR 15(1):1929–1958.
Xie, Z.; Avati, A.; Arivazhagan, N.; Jurafsky, D.; and Ng, A. Y. 2016. Neural language correction with character-based attention. arXiv preprint arXiv:1603.09727.
Yuan, Z., and Briscoe, T. 2016. Grammatical error correction using neural machine translation. In Proc. NAACL.