System	JFLEG Dev		JFLEG Test	
	$F_{0.5}$	GLEU	$F_{0.5}$	GLEU
MLConv$_{\text{embed}}$	56.67	47.71	58.82	51.34
MLConv$_{\text{embed}}$ (4 ens.)	57.62	47.93	59.21	51.06
+EO	59.40	49.24	62.15	53.38
+LM	62.44	51.24	65.87	55.99
+SpellCheck	63.61	52.48	66.80	57.47
C&N (**?**)	58.17	48.17	60.95	53.18
+SpellCheck	61.51	51.01	64.25	56.78

Table 2: Results on the JFLEG development and test sets compared to (**?**) (C&N). For MLConv$_{\text{embed}}$, average $F_{0.5}$ and GLEU of 4 models (trained with different random initializations) are reported.

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 $F_{0.5}$. After ensembling four models (4 ens.), the performance reaches 48.05 $F_{0.5}$ and outperforms the previous best neural model without LM (**?**) (41.53 $F_{0.5}$) by a large margin of 6.52 $F_{0.5}$, 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 rescoring is performed with edit operation (+EO) features, the performance goes up to 49.78 $F_{0.5}$, outperforming a strong SMT-based system (**?**) 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 (MLConv$_{\text{embed}}$), decode using an ensemble of four models, and rescore with edit operation features, the performance reaches 50.70 $F_{0.5}$.

After adding the web-scale LM in rescoring (+LM), our approach reaches 54.13 $F_{0.5}$, outperforming the best previous published result of (**?**) ($F_{0.5}$ = 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 (**?**) (+SpellCheck), our performance reaches 54.79, a statistically significant improvement of 1.65 $F_{0.5}$ ($p < 0.001$) 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 (**?**) 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 $F_{0.5}$ scores on these datasets. It is worth noting that our models achieve this level of performance without tuning on the JFLEG development set.

Architecture	Prec.	Recall	$F_{0.5}$
BiLSTM	52.49	10.95	29.84
SLConv	43.65	10.23	26.39
MLConv	51.90	12.59	31.96

Table 3: Performance of various architectures on the CoNLL-2013 test set.

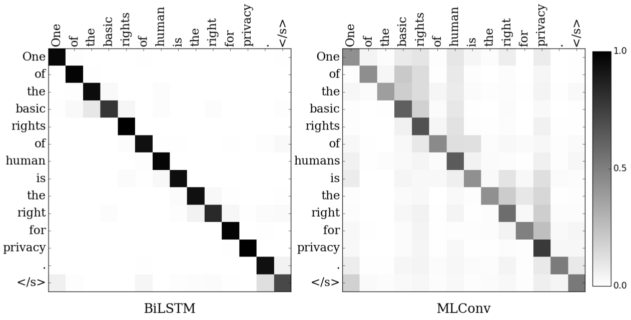

Figure 2: Visualization of attention weights of BiLSTM and MLConv models. y-axis shows the target words and x-axis shows the source words.

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 (**?**) (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 $\times$ 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

Initialization	Prec.	Recall	$F_{0.5}$
Random	51.90	12.59	31.96
Word2vec	52.80	12.80	32.49
fastText	51.08	13.63	32.97

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

ability of MLConv in capturing the context better, thereby favoring more corrections than copying of the source words.

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 (**?**), using the recently released ERRANT toolkit (**?**) on the CoNLL-2014 test data based on $F_{0.5}$. 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 ($\text{MLConv}_{\text{embed}}$) 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.

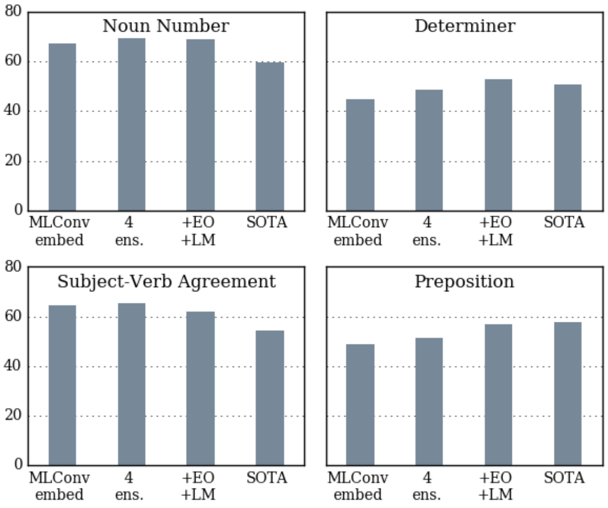

Figure 3: Performance of our models compared to the state-of-the-art system (**?**) on common error types evaluated on the CoNLL-2014 test set based on $F_{0.5}$.

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. Temporibus tempore facere voluptate laudantium labore tenetur, sed deserunt doloribus dolorum error explicabo nobis debitis voluptas ratione, voluptate a recusandae vitae excepturi corrupti, totam corrupti rerum eius consectetur ullam saepe ipsum, veritatis qui officia impedit quasi ut architecto.Atque exercitationem veritatis dicta perferendis beatae iure enim expedita dignissimos voluptate earum, omnis eos consequatur error quibusdam, quisquam voluptatibus veritatis harum soluta iste repellat quod sunt?Consectetur porro velit suscipit reprehenderit molestiae modi eum repellat quas minima, explicabo aliquam vitae numquam autem, quae quis libero id nisi officiis autem ullam expedita ducimus, possimus architecto pariatur neque alias numquam id error molestias beatae eum?Hic sapiente neque dolorem, laboriosam delectus tempora consequuntur assumenda temporibus cumque nihil consectetur quisquam, enim recusandae