Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data

Citation for published version:

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data

Roman Grundkiewicz† Marcin Junczys-Dowmunt‡ Kenneth Heafield†
† University of Edinburgh, Scotland, EU
‡ Microsoft, Redmond, USA
{rgrundki,kheafiel}@inf.ed.ac.uk, marcinjd@microsoft.com

Abstract

Considerable effort has been made to address the data sparsity problem in neural grammatical error correction. In this work, we propose a simple and surprisingly effective unsupervised synthetic error generation method based on confusion sets extracted from a spellchecker to increase the amount of training data. Synthetic data is used to pre-train a Transformer sequence-to-sequence model, which not only improves over a strong baseline trained on authentic error-annotated data, but also enables the development of a practical GEC system in a scenario where little genuine error-annotated data is available. The developed systems placed first in the BEA19 shared task, achieving 69.47 and 64.24 F$_{0.5}$ in the restricted and low-resource tracks respectively, both on the W&I+LOCNESS test set. On the popular CoNLL 2014 test set, we report state-of-the-art results of 64.16 M$^2$ for the submitted system, and 61.30 M$^2$ for the constrained system trained on the NUCLE and Lang-8 data.

1 Introduction

For the past five years, machine translation methods have been the most successful approach to automated Grammatical Error Correction (GEC). Work started with statistical phrase-based machine translation (SMT) methods (Junczys-Dowmunt and Grundkiewicz, 2016; Chollampatt and Ng, 2017) while sequence-to-sequence methods adopted from neural machine translation (NMT) lagged in quality until recently (Chollampatt and Ng, 2018; Junczys-Dowmunt et al., 2018b). These two papers established a number of techniques for neural GEC, such as transfer learning from monolingual data, strong regularization, model ensembling, and using a large-scale language model.

Subsequent work highlighted two challenges in neural GEC, data sparsity and multi-pass decoding:

Data sparsity: parallel training data has been enlarged by generating additional parallel sentences during training (Ge et al., 2018a,b), synthesizing noisy sentences (Xie et al., 2018), or pre-training a neural network on a large-scale but out-of-domain parallel corpus from Wikipedia (Lichtarge et al., 2018).

Multi-pass decoding: the correction process has been improved by incrementally correcting a sentence multiple times through multi-round inference using a model of one type (Ge et al., 2018a; Lichtarge et al., 2018), involving right-to-left models (Ge et al., 2018b), or by pipelining SMT and NMT-based systems (Grundkiewicz and Junczys-Dowmunt, 2018).

Motivated by the problems identified in these papers but concerned by the complexity of their methods, we sought simpler and more effective approaches to both challenges. For data sparsity, we propose an unsupervised synthetic parallel data generation method exploiting confusion sets from a spellchecker to augment training data used for pre-training sequence-to-sequence models. For multi-pass decoding, we use right-to-left models in rescoring, similar to competitive neural machine translation systems.

In the Building Educational Application (BEA) 2019 Shared Task on Grammatical Error Correction† (Bryant et al., 2019), our GEC systems ranked first in the restricted and low-resource tasks.² This confirms the effectiveness of the proposed methods in scenarios with and without readily-available large amounts of error-annotated data.

The rest of the paper is organized as follows:

†https://www.cl.cam.ac.uk/research/nl/bea2019st/
²Incidentally, our restricted system also outperformed all submissions to the unrestricted task to which we did not submit.
Section 2 briefly describes the BEA19 shared task and Section 3 presents related work. In Section 4 we demonstrate components of our neural GEC systems: transformer models, unsupervised synthetic data generation, ensembling and rescoring methods. Section 5 provides details of the experiments. The results are discussed in Sections 6 and 7, and we summarize in Section 8.

2 BEA19 shared task

The object of the BEA 2019 shared task was to automatically correct errors in written text, including grammatical, lexical, and orthographic errors. The shared task introduced two new annotated datasets for development and evaluation: Cambridge English Write & Improve (W&I) and the LOCNESS corpora (Bryant et al., 2019; Granger, 1998). These represent a more diverse cross-section of English language levels and domains than previous datasets.

There were three tracks that varied in the amount of admissible annotated learner data for system development. In the restricted track, participants were provided with four learner corpora containing 1.2 million sentences in total: the public FCE corpus (Yannakoudakis et al., 2011), NUCLE (Dahlmeier et al., 2013), Lang-8 Corpus of Learner English (Mizumoto et al., 2012), and the mentioned W&I+LOCNESS datasets. No restriction was placed on publicly available unannotated data or NLP tools such as spellcheckers. The low-resource track was limited to the use of the W&I+LOCNESS development set. The organizers further clarified that automatically extracted parallel data, e.g. from Wikipedia, could be used only to build low-resource and unrestricted systems; it was inadmissible in the restricted track. We participated in the restricted and low-resource tracks; the third track allowed unrestricted data.

The performance of participating systems was evaluated using the ERRANT scorer (Bryant et al., 2017) which reports a $F_{0.5}$ over span-based corrections.

3 Related work

Many recent advances in neural GEC aim at overcoming the mentioned data sparsity problem. Ge et al. (2018a) proposed fluency-boost learning that generates additional training examples during training from an independent backward model or the forward model being trained. Xie et al. (2018) supplied their model with noisy examples synthesized from clean sentences. Junczys-Dowmunt et al. (2018b) utilized a large amount of monolingual data by pre-training decoder parameters with a language model, and Lichtarge et al. (2018, 2019), on the other hand, used a large-scale out-of-domain parallel corpus extracted from Wikipedia revisions to pre-train their models. We also pre-train a neural sequence-to-sequence model, but we do so solely on synthetic data.

Although our unsupervised method for synthesising parallel data by means of an (inverted) spellchecker is novel, the idea of generating artificial errors has been explored in the literature before, as summarized by Felice (2016). Previously proposed methods usually require an error-annotated corpus as a seed to generate artificial errors reflecting linguistic properties and error distributions observed in natural-error corpora (Foster and Andersen, 2009; Felice and Yuan, 2014). Artificial error generation methods spanned conditional probabilistic models for specific error types only (Rozovskaya and Roth, 2010; Rozovskaya et al., 2014; Felice and Yuan, 2014), statistical or neural MT systems trained on reversed source and target sides (Rei et al., 2017; Kasewa et al., 2018) or neural sequence transduction models (Xie et al., 2018). None of these methods is unsupervised.

Other recent work focuses on improving model inference. Ge et al. (2018a) proposed correcting a sentence more than once through multi-round model inference. Lichtarge et al. (2018) introduced iterative decoding to incrementally correct a sentence with a high-precision system. The multi-round correction approach has been further extended (Ge et al., 2018b) by interchanging decoding of a standard left-to-right model with a right-to-left model. The authors claim that the two models display unique advantages for specific error types as they decode with different contexts. Inspired by this finding, we adapt a common technique from NMT (Sennrich et al., 2016, 2017) that reranks with a right-to-left model, but without multiple rounds. We contend that multiple rounds are only necessary if the system has low recall.

4 System overview

4.1 Transformer models

Our neural GEC systems are based on Transformer models (Vaswani et al., 2017) that have been recently adapted to grammatical error correction with
We apply GEC-specific adaptations proposed by Junczys-Dowmunt et al. (2018b) with some modifications. Following the paper, we use extensive regularization to avoid overfitting to the limited labelled data, including dropping out entire source embeddings (Sennrich et al., 2016), and additional dropout on attention and feed-forward network transformer layers. For the sake of simplicity, we replace averaging the best four model checkpoints with exponential smoothing (Gardner, 1985). We increase the size of mini-batches as this improved the performance in early experiments. Parameters of the full model are pre-trained on synthetic parallel data, instead of pre-training only the decoder parameters (Ramachandran et al., 2017). We also experiment with larger Transformer models as described in Section 5.3.

### 4.2 Synthetic data generation

Synthetic parallel training examples for GEC could be generated by substituting random words in an error-free sentence and using the pair of artificial and original sentences as a new training example. In a naïve approach, words can be replaced randomly within a vocabulary, but this may result in unrealistic error patterns that do not resemble those observed in the genuine data. More accurate errors can be generated by replacing words only within confusion sets if such a confusion set consists of words that are commonly confused with each other (Rozovskaya and Roth, 2010; Rozovskaya et al., 2014; Bryant and Briscoe, 2018).

Instead of applying a supervised probabilistic method to learn error distributions (Felice and Yuan, 2014; Rei et al., 2017; Xie et al., 2018; Kasewa et al., 2018; Bryant and Briscoe, 2018), we propose generating confusion sets with the help of a spellchecker. For each word in the vocabulary that consists of only alphabetic characters, including correct words, we extract suggestions from the Aspell spellchecker to create the confusion set of that word. Aspell sorts suggestion lists by a score that is the weighted average of the weighted edit distance of the proposed word to the input word and the distance between their phonetic equivalents generated by the metaphone algorithm (Philips, 2000). Confusion sets are limited to top 20 suggestions. Table 1 presents examples of generated confusion sets.

<table>
<thead>
<tr>
<th>Word</th>
<th>Confusion set</th>
</tr>
</thead>
<tbody>
<tr>
<td>has</td>
<td>Haas HS Hans hats gas had Ha ha As as</td>
</tr>
<tr>
<td>is</td>
<td>IRS ISO OS US us Si its</td>
</tr>
<tr>
<td>island</td>
<td>islands Iceland slant</td>
</tr>
<tr>
<td>issued</td>
<td>issues issue used issuers eased sued assumed assured missed</td>
</tr>
<tr>
<td>student</td>
<td>students strident stunt</td>
</tr>
<tr>
<td>walking</td>
<td>talking whaling</td>
</tr>
<tr>
<td>large</td>
<td>larger lag lake barge Lodge lodge</td>
</tr>
<tr>
<td>largest</td>
<td>latest longest</td>
</tr>
</tbody>
</table>

Table 1: Examples of confusion sets generated from a spellchecker.

Synthetic errors are introduced into an error-free text in the following manner. For each sentence, we sample an error probability $p_{err}$ from a normal distribution with mean 0.15, chosen to resemble the word error rate of the development set, and arbitrary standard deviation 0.2. This is multiplied by sentence length and rounded to a number of words to change. Exactly that many words in the sentence are chosen by sampling uniformly without replacement. Next, for each chosen word, we perform one of the following operations with a given probability: substituting $w_i$ with a random word from its confusion set, deleting $w_i$, inserting a random word after $w_i$, or swapping it with an adjacent word $w_{i+1}$. The probability for word substitution is set arbitrarily to 0.7 and the three remaining operations are chosen with a probability of 0.1 each.

Furthermore, to make our models more capable of correcting spelling errors, similarly to Lichtarge et al. (2018), we introduce additional noise in source words. We randomly perturb characters in 10% of words using the same operations as above for the word level operations, i.e. substitution, deletion, insertion or transposition of characters, with the same probabilities. An example of a synthetic sentence is presented in Table 2.

The proposed method does not generate context-aware errors, but is simple and can be applied to any alphabetic language with existing spell-checkers. In preliminary experiments, confusion sets generated using a spellchecker led to better performance during pre-training than methods based on the Levenshtein edit distance (Levenshtein, 1966) or word-

---

1 We add noise into the subword-segmented texts, so the vocabulary here is the same as the training vocabulary.

2 http://aspell.net/0.50-doc/man-html/8_How.html
But they have left their exam rooms and come out the streets to joining hands with the public and to fight for the country under the guidance of the monks.

But they have lifted their exam rooms end come out the streets to joining lands with the public band to fight for country the unity the guidance of the monos.

But they have lift their exam rooms end out the streets to joining lands with the public band to fight for country the unity the guidance of the monos.

Table 2: An example of an artificially generated erroneous sentence.

<table>
<thead>
<tr>
<th>Type</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original input</td>
<td>But they have left their exam rooms and come out the streets to joining hands with the public and to fight for the country under the guidance of the monks.</td>
</tr>
<tr>
<td>+ Synthetic errors</td>
<td>But they have lift their exam rooms end come out the streets to joining lands with the public band to fight for country the unity the guidance of the monos.</td>
</tr>
<tr>
<td>+ Spelling errors</td>
<td>But they have lift their exam rooms end out the streets to joining lands with the public band to fight for country the unity the guidance of the monos.</td>
</tr>
</tbody>
</table>

Table 3: Parallel and monolingual training data. R denotes datasets used to develop our restricted systems, L — low-resource systems.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Track</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCE Train</td>
<td>R</td>
<td>28,350</td>
</tr>
<tr>
<td>NUCLE</td>
<td>R</td>
<td>57,113</td>
</tr>
<tr>
<td>Lang-8</td>
<td>R</td>
<td>1,041,409</td>
</tr>
<tr>
<td>W&amp;I Train</td>
<td>R</td>
<td>34,308</td>
</tr>
<tr>
<td>W&amp;I+LOCNESS Dev</td>
<td>L,R</td>
<td>4,384</td>
</tr>
<tr>
<td>WikEd</td>
<td>L</td>
<td>2,000,000</td>
</tr>
<tr>
<td>News Crawl</td>
<td>L,R</td>
<td>100M</td>
</tr>
</tbody>
</table>

4.3 Model pre-training and fine-tuning

We generate synthetic errors from 100 million sentences sampled from the English part of the WMT News Crawl corpus (Bojar et al., 2018) and use pairs of synthetic and authentic sentences exclusively to pre-train transformer models. A pre-trained model can be used with the actual in-domain error-annotated data by fine-tuning (Hinton and Salakhutdinov, 2006; Miceli Barone et al., 2017). We experimented with two fine-tuning strategies:

1. Initialising the neural network weights with the pre-trained model and starting a new training run on new data. This resets learning rate scheduling and optimizer parameters. We further refer to this procedure as re-training.

2. Continuing training the existing model with new data preserving the learning rate, optimizer parameters and historic weights for exponential smoothing. We refer to this scheme as fine-tuning.

The main difference between re-training and fine-tuning is resetting the training state after pre-training. The latter strategy worked best in our experiments.

4.4 Ensembling

Similarly to Junczys-Dowmunt et al. (2018b), we build a heterogeneous ensemble of independently trained sequence-to-sequence models and a language model (LM). Sequence-to-sequence models are weighted equally, while the weight for the LM is grid-searched on the development set.

4.5 Right-to-left re-ranking

A common approach to improve the performance of NMT systems is re-ranking with right-to-left models that have been trained on the reversed word direction (Sennrich et al., 2016, 2017). In GEC, Ge et al. (2018b) use a right-to-left model for multi-round error correction where models following opposite sequence direction are run recursively one followed by another. The motivation is that both models use different contexts, so can be more capable of correcting errors of different types.

We adapt the re-ranking technique. We first generate n-best lists using the ensemble of standard left-to-right models and the language model, then re-score sentence pairs with right-to-left models using length-normalized scores, and re-rank the hypotheses. We have experimented with different weighting strategies during re-scoring, but found that weighting all sequence-to-sequence models equally with 1.0 and grid-searching the weight of the language model again works best. Tuning all ensemble weights independently with MERT (Och, 2003) lead to overfitting to the development set.

5 Experiments

5.1 Datasets

Error-annotated data The restricted models are trained on data provided in the shared task: the
FCE corpus (Yannakoudakis et al., 2011), NUCLE (Dahlmeier et al., 2013), W&I+LOCNESS data sets (Bryant et al., 2019; Granger, 1998), and a pre-processed version of the Lang-8 Corpus of Learner English (Mizumoto et al., 2012).

We clean Lang-8 using regular expressions to 1) filter out sentences with a low ratio of alphabetic to non-alphabetic tokens, 2) clear sentences from emoticons and sequences of repeated single non-alphanumeric characters longer than 3 elements e.g. repeated question or exclamation marks, and 3) remove trailing brackets with comments from the target sentences. If a sentence has alternative corrections, we expand them to separate training examples.

Our final training set in the restricted setting contains 1,953,554 sentences, assembled from the cleaned Lang-8 corpus and oversampled remaining corpora: FCE and the training portion of W&I are oversampled 10 times, NUCLE 5 times. Table 3 summarizes all data sets used for training. W&I+LOCNESS Dev is used solely as a development set in both tracks.

**Monolingual data** We use News Crawl — a publicly available corpus of monolingual texts extracted from online newspapers released for the WMT series of shared tasks (Bojar et al., 2018) — as our primary monolingual data source. We uniformly sampled 100 million English sentences from de-duplicated crawls in years 2007 to 2018 to produce synthetic parallel data for model pre-training. Another subset of 2 million sentences was selected to augment the training data during fine-tuning.

The Enchant spellchecker with the Aspell backend and a British English dictionary were used to generate confusion sets.

**Wikipedia edits** In the low-resource setting, we use a filtered subset of the WikEd corpus (Grundkiewicz and Junczys-Dowmunt, 2014). The original corpus contains 56 million automatically extracted edited sentences from Wikipedia revisions and is quite noisy.

We clean the data using cross-entropy difference filtering by Moore and Lewis (2010). W&I+LOCNESS Dev is used as an in-domain seed corpus. All sentence pairs in WikEd are sorted w.r.t an average score from two language models: an \( n \)-gram probabilistic word-level language model estimated from target sentences, and a simplified operation sequence model built on edits between source and target sentences. We use KenLM (Heafield, 2011) to build 5-gram language models. The top 2 million sentence pairs with the highest scores are used as training data in place of the error-annotated ESL learner data to train models for the low-resource system.

### 5.2 Data preprocessing

Following the preprocessing methods of the data provided in the shared task, we tokenize other data sets with spaCy. We also normalize Unicode punctuation to ASCII with a script included in the Moses SMT toolkit (Koehn et al., 2007).

To handle the open vocabulary issue, we split tokens into 32,000 subword units trained on 10 million randomly sampled sentences from News Crawl using the default unigram-LM segmentation algorithm (Kudo, 2018) from SentencePiece (Kudo and Richardson, 2018).

### 5.3 Model architecture

We experiment with different variants of Transformer models (Vaswani et al., 2017). The “Transformer Base” architecture has 6 blocks of self-attention/feed forward sub-layers in the encoder and decoder, 8-head self-attention layers, and embeddings vector size of 512. The ReLU activation function (Nair and Hinton, 2010) is used between filters of size 2048. We tie output layer, decoder and encoder embeddings (Press and Wolf, 2017).

We choose the “Transformer Big” architecture as our final models for the restricted track. They differ from Transformer Base by the number of heads in multi-head attention components (16 heads instead for 8), larger embeddings vector size of 1024 and filter size of 4096.

The architecture of the language models corresponds to the structure of the decoder of the sequence-to-sequence model, either Transformer Base or Big.

---

8 For example, a sentence pair (“I think that the public transport will always be in the future.”, “I think that public transport will always exist in the future.”) is first converted into the sequence “\texttt{<del> the <sub> be <to> exist”}, and then a standard \( n \)-gram probabilistic language model is built on such edit operation sequences.

9 http://data.statmt.org/news-crawl/

<table>
<thead>
<tr>
<th>System</th>
<th>Restricted systems</th>
<th>Low-resource systems</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>W&amp;I+LOCNESS</td>
<td>FCE</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>Baseline</td>
<td>47.1</td>
<td>30.2</td>
</tr>
<tr>
<td>Baseline + LM pretraining</td>
<td>47.2</td>
<td>30.9</td>
</tr>
<tr>
<td>Pre-training on synthetic data</td>
<td>43.2</td>
<td>10.6</td>
</tr>
<tr>
<td>→ Re-training</td>
<td>53.2</td>
<td>35.8</td>
</tr>
<tr>
<td>→ Fine-tuning</td>
<td><strong>54.8</strong></td>
<td>34.2</td>
</tr>
<tr>
<td>+ 2M synthetic data</td>
<td>56.1</td>
<td>34.8</td>
</tr>
<tr>
<td>Enclave Base × 8</td>
<td>58.4</td>
<td>34.8</td>
</tr>
<tr>
<td>+ LM Base</td>
<td>57.3</td>
<td>37.7</td>
</tr>
<tr>
<td>+ LM Big</td>
<td>56.9</td>
<td>38.9</td>
</tr>
<tr>
<td>+ RL rescoring Base × 4</td>
<td>57.7</td>
<td>38.3</td>
</tr>
<tr>
<td>Ensemble Big × 4 + LM Big</td>
<td>58.5</td>
<td>36.8</td>
</tr>
<tr>
<td>+ RL rescoring Base × 4</td>
<td>59.1</td>
<td>36.8</td>
</tr>
</tbody>
</table>

Table 4: Results for restricted and low-resource systems on W&I+LOCNESS Dev and FCE Test. Stars (*) indicate the submitted systems.

<table>
<thead>
<tr>
<th>Method</th>
<th>P</th>
<th>R</th>
<th>F₀.₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensemble × 4 + LM</td>
<td>58.5</td>
<td>36.8</td>
<td>52.30</td>
</tr>
<tr>
<td>→ Second pass</td>
<td>58.2</td>
<td>37.3</td>
<td>52.36</td>
</tr>
<tr>
<td>→ Round-way right-left</td>
<td>55.7</td>
<td>40.0</td>
<td>51.64</td>
</tr>
<tr>
<td>→ Iterative decoding</td>
<td>58.3</td>
<td>37.2</td>
<td>52.37</td>
</tr>
<tr>
<td>→ Right-left rescoring</td>
<td>59.1</td>
<td>36.8</td>
<td>53.00</td>
</tr>
</tbody>
</table>

Table 5: Comparison of different methods for inference optimization for the final restricted system on W&I+LOCNESS Dev.

5.4 Training settings

We train all models with the Marian toolkit\(^{11}\) (Junczys-Dowmunt et al., 2018a), and generally follow the configuration proposed by Junczys-Dowmunt et al. (2018b).

Transformer models are trained using Adam (Kingma and Ba, 2014) with a learning rate of 0.0003 and linear warm-up for the first 16k updates, followed by inverted squared decay. For the larger models, we decrease the learning rate to 0.0002 and warm-up to 8k first updates. We train with synchronous SGD (Adam) and dynamically sized mini-batches fitted into 48GB GPU RAM memory across 4 GPUs, accumulating gradients for 3 iterations before making an update (Bogoychev et al., 2018). This results in mini-batches consisting of ca. 2,700 sentences. The maximum length of a training sentence is limited to 150 subword units. Strong regularization via dropout (Gal and Ghahramani, 2016) is used to dissuade the model from simply copying the input: we use a dropout probability between transformer layers of 0.3, for transformer self-attention and filters of 0.1, and for source and target words of 0.3 and 0.1 respectively. For source and target words we dropout entire embedding vectors, not just single neurons. We also use label smoothing with a weight of 0.1, and exponential averaging of model parameters with a smoothing factor of 0.0001.

During fine-tuning, we use the the cross-entropy training objective with edits up-weighted by a factor of $\Lambda = 2$ (Junczys-Dowmunt et al., 2018b).

The model is validated every 5000 updates on W&I+LOCNESS Dev using the ERRANT $F₀.₅$ score. Models are trained with early stopping with a patience of 10. Pre-training is additionally limited to 5 epochs. We decode with beam search with a beam size of 12, and normalize scores for each hypothesis by sentence length. The checkpoint with the highest $F₀.₅$ score on the development set is selected as a final model.

Right-to-left models are trained with exactly the same settings, the only difference is the reversed word order in source and target sentences\(^{12}\) with no further data processing requirements.

\(^{11}\)https://marian-nmt.github.io/

\(^{12}\)Training right-to-left models is built into Marian and can be enabled with the \(-right-left\) option.
Language models are trained with the same settings as sequence-to-sequence models, but validated every 10,000 updates on the target side of the development set.

6 Results on the development set

Table 4 summarizes the results of the experiments on the W&I+LOCNESS Dev and FCE Test in the restricted and low-resource settings.

Restricted systems We compare our models to two Transformer-based baselines trained solely on the original error-annotated data without and with transfer learning from the language model. Surprisingly, for the restricted system, pre-training the decoder parameters (Baseline + LM pretraining) does not yield much improvement. A major improvement is achieved, however, by pre-training of the entire neural network on the synthetic data (Re-training).

The fine-tuning strategy generally leads to better results than re-training, mostly due to increased precision. Adding 2 million of synthetic sentences to the error-annotated data — resulting approximately in an 1:1 ratio of genuine and artificial training examples (Sennrich et al., 2017) — further improves the performance.

Ensembling eight Transformer models with a language model and re-ranking the n-best lists with four right-to-left models leads to consistent improvements. The quality of the language model is important as using a stronger language model (LM Big) generally improves the scores.

The systems with bigger models (Ensemble Big x 4 + LM Big) have a higher precision and thus perform better on both datasets. Interestingly, re-ranking using smaller and relatively weaker right-to-left Transformer Base models is still beneficial.

The final system with four Transformer Big models constitutes our submission to the restricted track for the official evaluation in the shared task.

Low-resource systems For the low-resource task, we follow the same experiments as for the restricted task, replacing the error-annotated training data with a subset of the filtered WikEd corpus of comparable size. Using out-of-domain data in place of the high-quality ESL learner data reduces the performance substantially in the low-resource baseline, but the gap is reduced in the final systems. Ensembling and re-ranking lead to larger relative improvements than for the restricted systems.

Due to a tight time frame, the final system submitted to the low-resource track uses eight Transformer Base models.

6.1 Proficiency levels and error types

The key contribution of the BEA19 shared task is the introduction of the W&I+LOCNESS dataset that consists of texts written by students of different English skill levels (A, B and C represents beginner, intermediate and advanced levels, respectively), including native texts (N). Figure 1 compares F₀.₅ scores of the corresponding restricted and low-resource ensemble systems for distinct parts of W&I+LOCNESS Dev.

Generally the higher the proficiency level of ESL texts, the lower the advantage of the systems trained on real error-annotated ESL learner data. Interestingly, the performance of restricted and low-resource systems on native texts is identical. It remains to be investigated if pre-training (the common part for those systems) is responsible for this.

As can be seen in Figure 2, the restricted and low-resource systems achieve similar performance on specific error types, for instance, morphology and subject-verb agreement errors, some errors within nouns, or misspellings.

6.2 Comparison to the state of the art

To compare with the current state of the art, we evaluate our best systems on other popular GEC benchmarks in Table 6. We report F₀.₅ scores on
the CoNLL 2014 test set (Dahlmeier et al., 2013) calculated with the official M2Scorer (Dahlmeier and Ng, 2012). We also report results on the JFLEG test set (Napoles et al., 2017) using GLEU (Napoles et al., 2015). Following other works (Sakaguchi et al., 2017; Junczys-Dowmunt et al., 2018b), we correct spelling errors in JFLEG using Enchant before decoding.

On CoNLL-2014, our best GEC system achieves 64.16 M$^2$, which is the highest score reported on this test set so far, including the systems trained on non-publicly available resources (Ge et al., 2018a,b). Although comparing to prior work, the improvement is impressive, our submitted system uses the public FCE corpus and the new W&I Train sets and should not be directly contrasted with systems trained on the NUCLE and Lang-8 corpora only. In contrastive experiments, we have trained a system with four Transformer Base models using the NUCLE and Lang-8 data from Junczys-Dowmunt et al. (2018b). That system achieves 61.30 F$_{0.5}$, which is the state-of-the-art result for a constrained GEC system, and it is comparable to the results reported by Ge et al. (2018b) for their system trained on non-public data. We expect even higher scores if our system would consist of larger Transformer models as in our submission.

### 7 Official results

The evaluation in the shared task was performed on the blind W&I+LOCNESS test set consisting of 350 student essays and 4,477 sentences. Excerpts of the official rankings are presented in Table 7.13

Our final GEC system achieves an official result of 69.47 F-score, which ranks it first among 21 systems participating in the main track. The top two systems perform significantly better than the remaining systems. We outperform the second system mainly due to higher recall and better performance on non-native parts of the test set: our system is +1.7 better on texts written by beginner English learners and -1.1 worse on native texts.

<table>
<thead>
<tr>
<th>System</th>
<th>CoNLL</th>
<th>JFLEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chollampatt and Ng (2018)</td>
<td>54.79</td>
<td>57.47</td>
</tr>
<tr>
<td>Junczys-Dowmunt et al. (2018b)</td>
<td>55.80</td>
<td>59.90</td>
</tr>
<tr>
<td>Grundkiewicz and Junczys-Dowmunt (2018)</td>
<td>56.25</td>
<td>61.50</td>
</tr>
<tr>
<td>Lichtarge et al. (2018)</td>
<td>58.30</td>
<td>62.40</td>
</tr>
<tr>
<td>Stahlberg et al. (2019)</td>
<td>58.40</td>
<td>58.63</td>
</tr>
<tr>
<td>Lichtarge et al. (2019)</td>
<td>60.40</td>
<td>63.3</td>
</tr>
<tr>
<td>Zhao et al. (2019)</td>
<td>61.15</td>
<td>61.00</td>
</tr>
<tr>
<td>Ge et al. (2018b)</td>
<td>61.34</td>
<td>62.42</td>
</tr>
<tr>
<td>Our low-resource system</td>
<td>52.44</td>
<td>58.07</td>
</tr>
<tr>
<td>Our restricted system</td>
<td>64.16</td>
<td>61.16</td>
</tr>
<tr>
<td>Constrained system</td>
<td>61.30</td>
<td>61.22</td>
</tr>
</tbody>
</table>

Table 6: Comparison with other works on the CoNLL-2014 and JFLEG test sets. The results for the constrained system are reported for best systems according to CoNLL-2013 and JFLEG Dev.

Full rankings with detailed results: https://www.cl.cam.ac.uk/research/nl/bea2019st/#results
Table 7: Official results for top 5 systems in the BEA19 shared task in the restricted (top) and low-resource (bottom) tracks. UEdin-MS is our submission.

<table>
<thead>
<tr>
<th>#</th>
<th>Team</th>
<th>P</th>
<th>R</th>
<th>F₀.₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UEdin-MS</td>
<td>72.28</td>
<td>60.12</td>
<td>69.47</td>
</tr>
<tr>
<td>2</td>
<td>Kakao&amp;Brain</td>
<td>75.19</td>
<td>51.91</td>
<td>69.00</td>
</tr>
<tr>
<td>3</td>
<td>LAIX</td>
<td>73.17</td>
<td>49.50</td>
<td>66.78</td>
</tr>
<tr>
<td>4</td>
<td>CAMB-CLED</td>
<td>70.49</td>
<td>55.07</td>
<td>66.75</td>
</tr>
<tr>
<td>5</td>
<td>Shuyao</td>
<td>70.17</td>
<td>55.39</td>
<td>66.61</td>
</tr>
</tbody>
</table>

(a) Restricted track.

<table>
<thead>
<tr>
<th>#</th>
<th>Team</th>
<th>P</th>
<th>R</th>
<th>F₀.₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UEdin-MS</td>
<td>70.19</td>
<td>47.99</td>
<td>64.24</td>
</tr>
<tr>
<td>2</td>
<td>Kakao&amp;Brain</td>
<td>63.06</td>
<td>46.30</td>
<td>58.80</td>
</tr>
<tr>
<td>3</td>
<td>LAIX</td>
<td>62.01</td>
<td>31.25</td>
<td>51.81</td>
</tr>
<tr>
<td>4</td>
<td>CAMB-CUED</td>
<td>55.58</td>
<td>38.03</td>
<td>50.88</td>
</tr>
<tr>
<td>5</td>
<td>UFAL</td>
<td>50.47</td>
<td>29.38</td>
<td>44.13</td>
</tr>
</tbody>
</table>

(b) Low-resource track.

Our low-resource GEC system is also ranked first among 9 participating teams achieving 64.24 F₀.₅ and outperforming the second best system significantly by +5.4. Interestingly, this system achieves the highest F-score of 72.25 on the part of the test set written by native speakers, comparing to the best result of 71.94 F₀.₅ by Kakao&Brain in the restricted track.

We did not submit a system to the unrestricted track, however our best system outperforms all systems in this track.

8 Summary

We presented an unsupervised synthetic error generation method based on confusion sets generated from an inverted spellchecker. With this method we increased the amount of training data for a grammatical error correction system. The generated synthetic parallel corpus was used to pre-train the sequence-to-sequence model and then fine-tuned on authentic data, which improved the performance of the adapted Transformer model in comparison to a model trained on authentic data alone. We also demonstrated the effectiveness of this approach in a scenario where little genuine error-annotated ESL learner data is available. Our final systems\(^{14}\) consist of ensembles of sequence-to-sequence Transformer models and a Transformer-based language model re-ranked with right-to-left models.

The presented GEC systems form our submissions to the BEA19 shared task as the UEdin-MS team. They are ranked first in the restricted and low-resource tracks achieving 69.47 and 64.24 F₀.₅ score on the W&I+LOCNESS test set respectively. On the popular CoNLL 2014 test set, we report state-of-the-art results of 64.16 M² for the best submitted system, and 61.30 M² for a system trained on the NUCLE and Lang-8 data.

Acknowledgments

This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (http: //www.csd3.cam.ac.uk/), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/P020259/1), and DiRAC funding from the Science and Technology Facilities Council (www.dirac.ac.uk).

References


\(^{14}\)Models, system configurations and outputs are available from https://github.com/grammatical/pretraining-bea2019


