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[1] (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:

[1] https://www.cl.cam.ac.uk/research/nl/bea2019st/

[2] Incidentally, 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 a 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

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

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

very good results (Junczys-Dowmunt et al., 2018b; Lichtarge et al., 2018).

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[3] 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[4] 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.

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-

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

[4]http://aspell.net/0.50-doc/man-html/8_How.html

Type	Output
Original input	*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 .*
+ Synthetic errors	*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 .*
+ Spelling errors	*But they have lift their exm rooms end out the streets to joining lands with the public band to fight for counrty the unity the guidance of the monos .*

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

embedding similarities (Mikolov et al., 2013).

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.

Corpus	Track	Sentences
FCE Train	R	28,350
NUCLE	R	57,113
Lang-8	R	1,041,409
W&I Train	R	34,308
W&I+LOCNESS Dev	L,R	4,384
WikEd	L	2,000,000
News Crawl	L,R	100M

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

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[5] 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[6] — 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[7] 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.[8] 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.[9] We also normalize Unicode punctuation to ASCII with a script included in the Moses SMT toolkit[10] (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.

[5]Cleaning Lang-8 led to minor improvements during the preliminary experiments when no pre-training was used.

[6]`http://data.statmt.org/news-crawl/`

[7]`https://abiword.github.io/enchant/`

[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 „`<del> the <sub> be <to> exist`", and then a standard n-gram probabilistic language model is built on such edit operation sequences.

[9]`https://spacy.io/`

[10]`https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/normalize-punctuation.perl`

	Restricted systems				Low-resource systems			
	W&I+LOCNESS			FCE	W&I+LOCNESS			FCE
System	P	R	$F_{0.5}$	$F_{0.5}$	P	R	$F_{0.5}$	$F_{0.5}$
Baseline	47.1	30.2	42.37	47.46	37.3	18.3	30.89	30.53
Baseline + LM pretraining	47.2	30.9	**42.69**	47.61	39.7	20.2	33.14	34.46
Pre-training on synthetic data	43.2	10.6	26.76	34.00	43.2	10.6	26.76	34.00
$\rightarrow$ Re-training	53.2	35.8	**48.44**	51.53	44.2	20.8	36.11	35.91
$\rightarrow$ Fine-tuning	**54.8**	34.2	48.92	52.69	49.6	21.0	38.97	41.79
+ 2M synthetic data	56.1	34.8	50.01	53.64	53.6	18.9	39.16	42.23
Ensemble Base×8	58.4	34.8	51.42	53.92	55.0	20.8	41.37	43.75
+ LM Base	57.3	37.7	51.88	53.33	51.1	26.5	43.11	44.04
+ LM Big	56.9	38.9	52.04	53.17	52.0	26.7	43.69	45.68
+ RL rescoring Base×4	57.7	38.3	52.42	55.03	55.0	26.0	**44.95***	47.42
Ensemble Big×4 + LM Big	58.5	36.8	52.30	54.57			—	
+ RL rescoring Base×4	59.1	36.8	**53.00***	55.81			—	

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

Method	P	R	$F_{0.5}$
Ensemble×4 +LM	58.5	36.8	52.30
$\rightarrow$ Second pass	58.2	37.3	52.36
$\rightarrow$ Round-way right-left	55.7	40.0	51.64
$\rightarrow$ Iterative decoding	58.3	37.2	52.37
$\rightarrow$ Right-left rescoring	59.1	36.8	53.00

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 $F_{0.5}$ 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 $F_{0.5}$ 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$\times$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. We have found that re-ranking works best for our high-recall system, better than other methods for multi-pass decoding as presented in Table 5.

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.

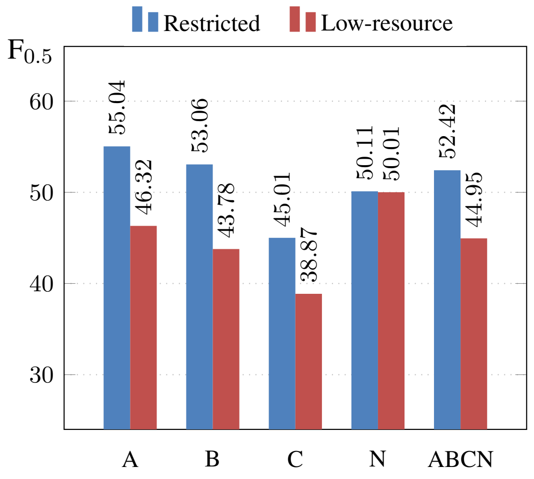

Figure 1: Comparison of restricted and low-resource systems on different parts of W&I+LOCNESS Dev.

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 $F_{0.5}$ 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-resourcse 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 $F_{0.5}$ scores on

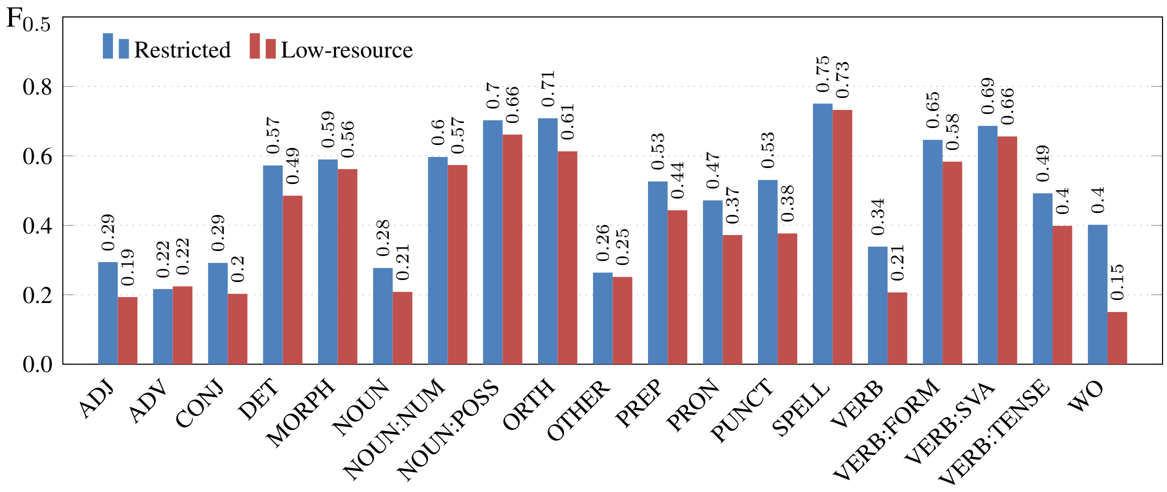

Figure 2: Comparison of restricted and low-resource systems ($F_{0.5}$) on a selection of error types from ERRANT on W&I+LOCNESS Dev.

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]

System	CoNLL	JFLEG
Chollampatt and Ng (2018)	54.79	57.47
Junczys-Dowmunt et al. (2018b)	55.80	59.90
Grundkiewicz and Junczys-Dowmunt (2018)	56.25	61.50
Lichtarge et al. (2018)	58.30	62.40
Stahlberg et al. (2019)	58.40	58.63
Lichtarge et al. (2019)	60.4	63.3
Zhao et al. (2019)	61.15	61.00
Ge et al. (2018b)	61.34	62.42
Our low-resource system	52.44	58.07
Our restricted system	64.16	61.16
Constrained system	61.30	61.22

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.

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.

[13]Full rankings with detailed results: `https://www.cl.cam.ac.uk/research/nl/bea2019st/#results`

#	Team	P	R	$F_{0.5}$
1	UEdin-MS	72.28	60.12	69.47
2	Kakao&Brain	75.19	51.91	69.00
3	LAIX	73.17	49.50	66.78
4	CAMB-CLED	70.49	55.07	66.75
5	Shuyao	70.17	55.39	66.61

(a) Restricted track.

#	Team	P	R	$F_{0.5}$
1	UEdin-MS	70.19	47.99	64.24
2	Kakao&Brain	63.06	46.30	58.80
3	LAIX	62.01	31.25	51.81
4	CAMB-CUED	55.58	38.03	50.88
5	UFAL	50.47	29.38	44.13

(b) Low-resource track.

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.

Our low-resource GEC system is also ranked first among 9 participating teams achieving 64.24 $F_{0.5}$ 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 $F_{0.5}$ 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 $F_{0.5}$ 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 M^2 for the best submitted system, and 61.30 M^2 for a system trained on the NUCLE and Lang-8 data.

[14]Models, system configurations and outputs are available from https://github.com/grammatical/pretraining-bea2019

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

Nikolay Bogoychev, Marcin Junczys-Dowmunt, Kenneth Heafield, and Alham Fikri Aji. 2018. Accelerating asynchronous stochastic gradient descent for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP'18, pages 2991–2996, Brussels, Belgium.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Christopher Bryant and Ted Briscoe. 2018. Language model based grammatical error correction without annotated training data. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 247–253, New Orleans, Louisiana. 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 14th Workshop on Innovative Use of NLP for Building Educational Applications*. 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. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 327–333, Copenhagen, Denmark. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

Mariano Felice. 2016. Artificial error generation for translation-based grammatical error correction. Technical report, University of Cambridge, Computer Laboratory.

Mariano Felice and Zheng Yuan. 2014. Generating artificial errors for grammatical error correction. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–126, Gothenburg, Sweden. Association for Computational Linguistics.

Jennifer Foster and Øistein E Andersen. 2009. Generrate: generating errors for use in grammatical error detection. In *Proceedings of the fourth workshop on innovative use of nlp for building educational applications*, pages 82–90. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Everette Gardner. 1985. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1):1–28.

Tao Ge, Furu Wei, and Ming Zhou. 2018a. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1055–1065, Melbourne, Australia. Association for Computational Linguistics.

Tao Ge, Furu Wei, and Ming Zhou. 2018b. Reaching human-level performance in automatic grammatical error correction: An empirical study. Technical report, Microsoft Research Technical Report.

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

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2014. The WikEd error corpus: A corpus of corrective Wikipedia edits and its application to grammatical error correction. In *Advances in Natural Language Processing – Lecture Notes in Computer Science*, volume 8686, pages 478–490. 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. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 187–197, Stroudsburg, USA. Association for Computational Linguistics.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, Austin, Texas. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018a. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018b. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606, New Orleans, Louisiana. Association for Computational Linguistics.

Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. 2018. Wronging a right: Generating better errors to improve grammatical error detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4977–4983, Brussels, Belgium. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. 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 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1494, Copenhagen, Denmark. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yu Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012*, pages 863–872.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden. Association for Computational Linguistics.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

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, Beijing, China. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 2017 Conference of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, USA. Association for Computational Linguistics.

Lawrence Philips. 2000. The double metaphone search algorithm. *C/C++ Users J.*, 18(6):38–43.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391, Copenhagen, Denmark. Association for Computational Linguistics.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 287–292, Copenhagen, Denmark. Association for Computational Linguistics.

A. Rozovskaya, D. Roth, and V. Srikumar. 2014. Correcting grammatical verb errors. In *European Chapter of the Association for Computational Linguistics*.

Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 961–970, Cambridge, MA. Association for Computational Linguistics.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 366–372, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399, Copenhagen, Denmark. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Felix Stahlberg, Christopher Bryant, and Bill Byrne. 2019. Neural grammatical error correction with finite state transducers. 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 4033–4039, Minneapolis, Minnesota. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 619–628, New Orleans, Louisiana. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

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. Association for Computational Linguistics.