Multi-Class Grammatical Error Detection for Correction: A Tale of Two Systems

Zheng Yuan Shiva Taslimipoor Christopher Davis Christopher Bryant
ALTA Institute
Department of Computer Science and Technology
University of Cambridge, United Kingdom
{firstname.lastname}@cl.cam.ac.uk

Abstract

In this paper, we show how a multi-class grammatical error detection (GED) system can be used to improve grammatical error correction (GEC) for English. Specifically, we first develop a new state-of-the-art binary detection system based on pre-trained ELECTRA, and then extend it to multi-class detection using different error type tagsets derived from the ERRANT framework. Output from this detection system is used as auxiliary input to fine-tune a novel encoder-decoder GEC model, and we subsequently re-rank the N-best GEC output to find the hypothesis that most agrees with the GED output. Results show that fine-tuning the GEC system using 4-class GED produces the best model, but re-ranking using 55-class GED leads to the best performance overall. This suggests that different multi-class GED systems benefit GEC in different ways. Ultimately, our system outperforms all other previous work that combines GED and GEC, and achieves a new single-model NMT-based state of the art on the BEA-test benchmark.

1 Introduction

Grammatical error detection (GED) is the task of automatically detecting grammatical errors in text, while grammatical error correction (GEC) is the task of also correcting these errors. Both tasks have obvious pedagogical applications that can benefit both teachers and students in online language learning. GED is typically cast as a binary sequence labelling task, where each token is classified as either correct or incorrect (Rei and Yannakoudakis, 2016; Bell et al., 2019), while GEC is often considered a sequence-to-sequence translation task, where systems learn to "translate" an ungrammatical input sentence to a grammatical output sentence (Yuan and Briscoe, 2016; Junczys-Dowmunt et al., 2018; Kiyono et al., 2019; Lichtarge et al., 2020). Recent work has also begun to treat GEC as a sequence labelling task, where tokens are classified in terms of edit operations (Awasthi et al., 2019; Omelianchuk et al., 2020). We similarly treat GED as a sequence labelling task and GEC as a sequence-to-sequence task, but additionally investigate different ways to combine and extend both approaches.

In particular, we first experiment with pre-trained language models, and show that simply fine-tuning ELECTRA (Clark et al., 2020) leads to significant improvements in binary GED and achieves a new state of the art. Given that binary detection is limited in terms of the specific error type information it can provide to downstream tasks however, we also extend our GED system to 4-class, 25-class and 55-class error detection using different error type tagsets derived from the ERRANT framework (Bryant et al., 2017).

To illustrate the value of multi-class GED for downstream GEC, we extend the Transformer encoder-decoder model (Vaswani et al., 2017) and employ a multi-encoder GEC model (Yuan and Bryant, 2021). We also introduce a two-step training strategy which only requires additional input information from a small dataset for fine-tuning. Specifically, we experiment with two methods that use GED to inform GEC: i) we use GED predictions as auxiliary input to fine-tune a model; and ii) we use GED predictions as a means to re-rank system output during post-processing.

To summarise, we present the first study using multi-class GED to improve GEC for English. Our main contributions are:

- We obtain a new state of the art in binary GED on three benchmark datasets.
- We empirically show that current Transformer-based language models are capable of much more fine-grained error detection, with minimal impact to overall binary $F_{0.5}$.
- We propose a novel multi-encoder GEC model and two-step training strategy, which has proven to be effective at incorporating an

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 8722–8736
November 7–11, 2021.

additional GED signal.

- We demonstrate how multi-class GED can improve a GEC model by i) using GED predictions during fine-tuning; and ii) using GED predictions as a basis for re-ranking.
- We report competitive performance with the state of the art using a single model without task-specific adaptation.

2 Previous work

Early approaches to GED focused on specific error types, and in particular article and preposition errors, which are among the most frequent in non-native English learner writing (Han et al., 2004; Tetreault and Chodorow, 2008). More general open-class GED systems were later developed using parse and text-based features (Gamon, 2011). Rei and Yannakoudakis (2016) presented the first work using a neural approach and framed GED as a binary sequence labelling problem, classifying each token in a sentence as either correct or incorrect. Subsequent improvements were obtained through auxiliary objectives (Rei and Yannakoudakis, 2017; Rei, 2017), and incorporating artificial training data (Rei et al., 2017; Kasewa et al., 2018). Recent work takes advantage of large scale pre-trained language models: Bell et al. (2019) used pre-trained contextual embeddings from BERT (Devlin et al., 2019) as input to a bi-directional LSTM (Hochreiter and Schmidhuber, 1997), while Kaneko and Komachi (2019) utilised information from intermediate layers via a multi-head multi-layer attention architecture.

Beyond the development of machine learning classifiers for specific error types (De Felice and Pulman, 2008; Rozovskaya and Roth, 2011), GEC has been formulated as a monolingual machine translation task that corrects all error types simultaneously. Both statistical machine translation (SMT) and neural machine translation (NMT) have been successfully applied to GEC with various task-specific adaptations (Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2016; Yuan and Briscoe, 2016; Junczys-Dowmunt et al., 2018; Yuan et al., 2019). With recent advances in sequence-to-sequence modelling and the introduction of the Transformer architecture, state-of-the-art results have been reported (Kaneko et al., 2020; Lichtarge et al., 2020). Meanwhile, sequence-tagging approaches have been proposed for GEC, where systems learn to predict a sequence of edit operations (Awasthi et al., 2019; Omelianchuk et al., 2020). In fact Omelianchuk et al. (2020) achieved the current state of the art by incorporating pre-trained Transformer encoders and employing iterative tagging.

Previous work has attempted to combine these two similar tasks and explore different ways of using GED in GEC. Zhao et al. (2019) and Yuan et al. (2019) employed multi-task learning and introduced token-level and sentence-level GED as auxiliary tasks when training for GEC. Kaneko et al. (2020) fine-tuned BERT for binary GED and then incorporated their model into an encoder-decoder GEC framework. Similarly, Chen et al. (2020) fine-tuned RoBERTa (Liu et al., 2019) for GED and reformatted the input to include error span information, which was used by their encoder-decoder model. Our approach of using additional GED input during GEC training differs from theirs in that we only use a small set of training examples with GED information for fine-tuning. The GED training and GEC training are not coupled together, so GED predictions from any system can be used.

Binary GED has also been used in post-processing to re-rank GEC system output (Yannakoudakis et al., 2017; Yuan et al., 2019; Wang et al., 2020) or filter out unnecessary corrections (Kiyono et al., 2019). Similarly, Chollampatt and Ng (2018) and Liu et al. (2021) applied quality estimation approaches to re-rank GEC output. Additional GED and/or GEC features are often used in these systems however, which also require extra tuning. In contrast, our simple re-ranking approach only uses GED predictions and does not require tuning. Our work differs most centrally in that we treat GED as a multi-class problem and investigate ways to use multi-class GED predictions to inform GEC.

3 Data

Following previous studies, we use the public Lang-8 Corpus (Mizumoto et al., 2011; Tajiri et al., 2012), the First Certificate in English (FCE) corpus (Yannakoudakis et al., 2011), the Cambridge Learner Corpus (CLC) (Nicholls, 2003), the National University of Singapore Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013) and the Cambridge English Write & Improve + LOCNESS (W&I) corpus released in the Building Educational Applications (BEA) shared task on GEC (Bryant

Original	I	spend	quite	long	time	for	shopping	.
Binary	C	I	C	I	C	I	C	C
4-class	C	R	C	M	C	U	C	C
25-class	C	VERB:TENSE	C	DET	C	PREP	C	C
55-class	C	R:VERB:TENSE	C	M:DET	C	U:PREP	C	C
Corrected	I **spent** quite **a** long time shopping.							

Table 1: An example sentence with different error type labels in different multi-class settings.

et al., 2019).[1] Unlike current state-of-the-art systems, we do not use any native or artificial data.

The edit annotations in all these corpora were pre-processed and standardised using the ERRANT annotation framework. One advantage of this framework is that error types are modular, and consist of "operation" + "main" type tags; e.g. `R:NOUN` for replacement noun. We hence use this modularity in our multi-class GED experiments such that 4-classes consist of operation type only (i.e. missing, replacement, unnecessary and correct), 25-classes consist of main type only (e.g. noun, noun number, verb tense, etc.) and 55-classes consist of the full tags combined.[2] An example is shown in Table 1.

4 Grammatical error detection

Following Rei and Yannakoudakis (2016), we treat GED as a sequence labelling (or token classification) task and assign a label to each token in the input sentence, indicating whether it is correct or incorrect (i.e. binary classification) or which error type it belongs to at different levels of granularity (i.e. multi-class classification). We first perform binary classification to compare our models with current state-of-the-art systems. We further extend them with multi-class error classification with the aim of improving downstream GEC.

4.1 Approach

We employ three state-of-the-art pre-trained language representation models: BERT (Devlin et al., 2019), XLNet (Yang et al., 2019) and ELECTRA (Clark et al., 2020). Although different in their pre-training architectures, they are all Transformer-based models on top of which we add a linear classification layer. We fine-tune these models on annotated GED data for a small number of epochs. BERT is one of the most popular and pioneering transfer learning methods involving self-attention layers of encoders and decoders for which the pre-trained model weights are available. XLNet aims to overcome some shortcomings of BERT by using an auto-regressive architecture which relies on permutation rather than masking during pre-training. ELECTRA is an extension of BERT with a different pre-training task which is a discriminator (rather than a generator) and aims to detect replaced tokens. Intuitively, its objective to discriminate between plausible and non-plausible word tokens makes it more closely-related to GED.

4.2 Error detection experiments

All datasets contain manually annotated spans of various types of errors, together with their suggested corrections. We convert these spans into token-based labels, assigning missing word labels to the token on the right of the span. This is consistent with previous work and necessary because missing words fall between tokens and would otherwise not be represented. For binary detection, each token is labelled as either correct 'C' or incorrect 'I'. For multi-class detection, we use ERRANT error types at different levels of granularity (4-class, 25-class and 55-class) as described in Section 3 and exemplified in Table 1. We perform fine-tuning using the Adam optimiser with a learning rate of 3×10^{-5} for all three models.[3]

For binary GED, we follow Rei and Yannakoudakis (2016) and report token-level precision, recall and $F_{0.5}$ for detecting incorrect labels. For multi-class GED, we report 1) binarised $F_{0.5}$ which is the score for detecting any non-C labels regardless of class, and 2) macro-averaged $F_{0.5}$ which is the average $F_{0.5}$ across all classes.

[1] Detailed corpus statistics are given in Appendix A. Public data is available at: https://www.cl.cam.ac.uk/research/nl/bea2019st#data

[2] See Appendix A in Bryant et al. (2017) for all combinations.

[3] We use the pre-trained large models for BERT, XLNet, and ELECTRA provided by Hugging Face (https://huggingface.co).

System	BEA-dev			FCE-test			CoNLL-2014 Test 1			CoNLL-2014 Test 2		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
GED (BERT)	65.48	42.85	59.23	75.73	47.98	67.88	49.73	34.23	45.60	64.52	32.33	53.80
GED (XLNet)	70.03	45.55	63.23	77.50	49.81	69.75	53.23	36.17	48.64	70.68	34.95	58.68
GED (ELECTRA)	72.81	46.85	**65.54**	82.05	50.49	**72.93**	55.15	39.78	**51.19**	76.44	40.13	**64.73**
Bell et al. (2019)	53.31	35.65	48.50	64.96	38.89	57.28	38.04	33.12	36.94	53.52	30.05	46.29
Kaneko and Komachi (2019)	-	-	-	68.87	43.45	61.65	35.74	33.50	35.26	46.45	35.47	43.74

Table 2: Binary error detection performance on BEA-dev, FCE-test and CoNLL-2014; Test 1 and Test 2 refer to the two different CoNLL annotations. All systems are trained on the FCE.

Binary GED To be comparable with previous work (e.g. Bell et al. 2019; Kaneko and Komachi 2019), we first fine-tune our models for binary detection on the FCE dataset, and report results on BEA-dev, FCE-test and CoNLL-2014 (Ng et al., 2014). Results are presented in Table 2. As can be seen, all our Transformer-based GED models outperform the current state-of-the-art systems on all test sets by large margins, with fine-tuned ELECTRA performing the best overall. We believe that this is due to its intuitively more GED-relevant discriminative pre-training objective.

Multi-class GED Given that ELECTRA performed the best at binary GED, we use it in our multi-class GED experiments. Table 3 shows the binarised and macro-averaged $F_{0.5}$ scores for different binary and multi-class GED systems. As expected, we see lower macro-averaged scores for multi-class classification when there are a higher number of classes. This is due to the sparsity of the labels when we add more error types. It is interesting to note, however, that adding more error types does not significantly affect the performance of the models in terms of binarised detection. For example, the binarised performance of the 4-class and 55-class models is slightly higher than the binary model on BEA-dev (66.10 and 65.81 vs. 65.54). This may suggest that all systems are capable of detecting roughly the same number of errors despite the number of classes and generally struggle only with the specific class labels themselves.

5 Using GED to improve GEC

In this section, we investigate different ways of using GED to inform GEC. We use the Transformer sequence-to-sequence model as our baseline and employ a multi-encoder GEC system that takes additional GED predictions as input. We then experiment with two methods of using GED information: i) as auxiliary input, and ii) for re-ranking.

Mode	BEA-dev $F_{0.5}$		FCE-test $F_{0.5}$	
	binarised	macro	binarised	macro
binary	65.54	80.39	72.93	83.54
4-class	**66.10**	67.07	72.57	70.95
25-class	63.08	47.28	72.08	54.59
55-class	65.81	32.99	**73.85**	34.88

Table 3: Binary and multi-class error detection performance of the ELECTRA GED model trained on the FCE. The highest binarised $F_{0.5}$ scores are in bold.

5.1 Baseline GEC system

The Transformer follows an encoder-decoder architecture. Each layer of the encoder contains two sub-layers: a multi-head self-attention mechanism and a feed-forward network. The decoder inserts an additional third sub-layer, which performs multi-head attention over the output of the encoder stack. See Vaswani et al. (2017) for more details.

5.2 Multi-encoder GEC system

In order to incorporate GED into GEC, we propose a new extension to the standard Transformer encoder-decoder model, which employs a second encoder to take additional GED input (Figure 1).

Encoder The original Transformer encoder reads the source sentence S_{src} and learns a vector representation c_{src} as before. An additional encoder is introduced to process any auxiliary GED input S_{ged} and compute a context representation c_{ged}.

Decoder Similar to the original Transformer decoder, each layer of the decoder in our model is composed of three sub-layers. The first sub-layer performs masked multi-head self-attention on the known outputs at positions less than i. The second sub-layer now contains two multi-head attention components: a source multi-head attention (MH_{src}), which performs multi-head attention over the output of the encoder stack for the source sentence c_{src}, and a new GED multi-head attention

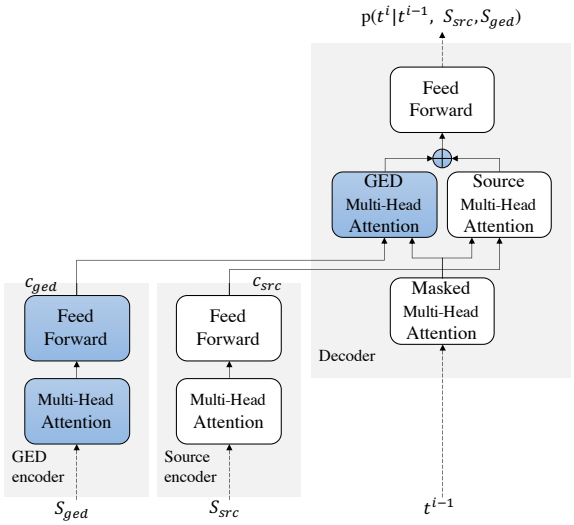

Figure 1: Multi-encoder GEC system. The newly introduced modules are highlighted in blue.

(MH_{ged}), that attends directly to the GED encoder representation c_{ged}. Afterwards, a linear interpolation with gating is applied:

$$\text{Gating(MH)} = \lambda \text{MH}_{src} + (1 - \lambda)\text{MH}_{ged} \quad (1)$$

The gating activation λ is given by:

$$\lambda = \sigma(W\,[\text{MH}_{src}; \text{MH}_{ged}] + b) \quad (2)$$

where σ is the logistic sigmoid function, and W and b are learnable parameters. The resulting Gating(MH) is used as an input to the third sub-layer, which is a position-wise fully connected feed-forward network.

Two-step training Inspired by the idea of freezing some parameters while fine-tuning the remaining part of the model (Zhang et al., 2018; Yuan and Bryant, 2021), we apply a two-step training strategy to train our proposed model. We divide model parameters into two subsets:

$$\theta = [\theta_{src}; \theta_{ged}] \quad (3)$$

where θ_{src} is a set of original Transformer model parameters, and θ_{ged} is a set of newly-introduced GED component parameters (i.e. the GED encoder, the GED multi-head attention and the gating - highlighted in blue in Figure 1).

In the first step, we train a standard encoder-decoder GEC model using standard source-target parallel data:

$$\hat{\theta}_{src} = \underset{\theta_{src}}{\text{argmax}} \sum_{n=1}^{N} logP(T^n|S^n, \theta_{src}) \quad (4)$$

where N is the total number of training examples and (S^n, T^n) is the nth source-target sentence pair.

In the second step, we construct a new dataset by adding GED information (S_{ged}) for each source-target pair and estimate GED parameters θ_{ged}:

$$\hat{\theta}_{ged} = \underset{\theta_{ged}}{\text{argmax}} \sum_{n=1}^{M} logP(T^n|S^n, S^n_{ged}, \hat{\theta}_{src}, \theta_{ged}) \quad (5)$$

where M is the total number of training examples in the new fine-tuning dataset ($M < N$) and (S^n, S^n_{ged}, T^n) is the nth triplet.

Our training strategy is different from most fine-tuning approaches in GEC (e.g. Kiyono et al., 2019; Lichtarge et al., 2020) as we only update θ_{ged} and keep θ_{src} fixed when adding GED auxiliary input. This is to prevent overfitting, as the dataset used in the second step is much smaller than the one used in the first step.

5.3 Using GED as auxiliary input

Experimental setup Following previous work that advocates fine-tuning GEC models on high-quality, in-domain data (Kiyono et al., 2019; Lichtarge et al., 2020; Yuan and Bryant, 2021), we pre-train two GEC systems on public Lang-8 data (constrained) and private CLC data (unconstrained) respectively, and fine-tune both on W&I + FCE + NUCLE. The constrained system thus only uses public data released in the BEA-2019 shared task. Two GED systems are similarly fine-tuned using the Lang-8 and CLC pre-training data,[4] and are used to make predictions on the W&I + FCE + NUCLE fine-tuning data that is auxiliary input to the GEC system.[5]

We use byte pair encoding (BPE) (Sennrich et al., 2016) with 30k merge operations. For both the Transformer baseline and our multi-encoder GEC models, the hidden size is set to 512 and the filter size is set to 2,048. In training, we use the Adam optimizer (Kingma and Ba, 2015) with the default parameters. Batch size is 3k tokens. All other settings follow the 6-layer 'Transformer (base)' model in Vaswani et al. (2017). We use four Tesla P40 GPUs for training and one for decoding. Experiments are carried out with Fairseq (Ott et al., 2019).

Results GEC performance on BEA-dev (evaluated by ERRANT) in the unconstrained setting is

[4]Performance shown in Appendix B.

[5]This setup was chosen because it would be unfair to train the GED system on the same data that is used as predicted input to the GEC system.

reported in Table 4. For the baseline model, we use the same pre-training and fine-tuning data splits, but with no additional GED input for fine-tuning, which follows the standard encoder-decoder GEC training procedure. For our new multi-encoder GEC model, we experiment with predictions from both the binary and multi-class GED models introduced in Section 4. The results demonstrate the efficacy of the multi-encoder GEC model: adding GED predictions as auxiliary input yields a consistent statistically significant improvement in performance over the baseline.[6] Our best system uses the 4-class GED predictions, achieving 52.84 $F_{0.5}$, followed by binary (52.20), 25-class (51.78) and 55-class (51.52). We suspect the 4-class system works best here because it represents the best compromise between label informativeness and model reliability. In contrast, binary predictions tend to be less informative but more reliable because there are only two classes, while 25-class and 55-class predictions tend to be more informative but less reliable because of the increased difficulty in predicting sparser classes. These results nevertheless show that multi-class GED provides GEC with new information, and we expect further performance gains with better multi-class GED systems. We also notice that almost all the improvements come from recall, with only a negligible influence on precision.

Finally, we also analysed the performance of each operation error type in the 4-class GEC system and found that most gains come from missing word errors (+6.66 $F_{0.5}$), which happen to be the worst performing type in the baseline system.[7]

Oracle To better understand our model's capacity to use GED information, we estimate an upper bound for the GEC system by using gold multi-class detection labels as auxiliary input. This provides us with the maximum performance our multi-encoder GEC system can obtain given a perfect GED system. In Table 4, we see that our system benefits the most when the finest and most granular level of error type information is provided. Specifically, results show that the maximum attainable score is achieved by the 55-class oracle GED system (70.24 $F_{0.5}$), followed by the 25-class (68.36 $F_{0.5}$), 4-class (67.86 $F_{0.5}$) and binary systems (63.68 $F_{0.5}$). This further supports the idea that the main bottleneck in a practical system is the reliability of the GED predictions rather than the informativeness of the labels. We finally observe only a small difference between the 4-class and 25-class oracle GED labels, suggesting that the 4-class operation labels (i.e. missing, replacement, unnecessary and correct) are about as informative as the 25-class POS-based error types for our multi-encoder GEC system.

		P	R	$F_{0.5}$
Baseline	w/o GED	58.52	31.20	49.80
Binary	+ GED	58.32	36.76	52.20
	+ *Oracle*	64.29	61.36	63.68
4-class	+ GED	58.68	37.78	**52.84**
	+ *Oracle*	68.37	65.86	67.86
25-class	+ GED	58.54	35.42	51.78
	+ *Oracle*	68.56	67.56	68.36
55-class	+ GED	57.50	36.38	51.52
	+ *Oracle*	70.49	69.27	**70.24**

Table 4: ERRANT span-level correction results on BEA-dev in the unconstrained setting of our multi-encoder GEC system using different GED models and oracle detection. The highest GED and oracle results are in bold.

5.4 Using GED for re-ranking

The GEC decoder generates different hypothesis sentences from which the sentence with the highest confidence score is predicted as the correction. Inspired by Yannakoudakis et al. (2017) and Yuan et al. (2019), we take advantage of these hypotheses and employ a re-ranking approach using GED outputs to further improve our GEC results. Specifically, we i) generate a 10-best list of candidate hypotheses for each sentence, ii) align each hypothesis with the source sentence using ERRANT to extract the edits, and iii) convert the edit spans to token-based detection labels as described in Section 4.2. This produces a list of hypotheses, where each hypothesis $H_{i\in\{1:10\}} = (h_{i,1}, h_{i,2}, ..., h_{i,l})$ consists of the error-type labels for each token in a source sentence of length l as predicted by the GEC system. We then use a GED system to predict a corresponding set of labels $D = (d_1, d_2, ..., d_l)$ for each source token, and re-rank the hypotheses based on the minimum Hamming distance between H_i and D. This ensures the maximal overlap between the GEC hypothesis and the (multi-class) GED predictions and hence provides greater confidence that a hypothesis is correct when more error-type labels from both systems agree.

[6] We perform two-tailed paired T-tests, where $p < 0.001$.

[7] Error analysis results are included in Appendix C.

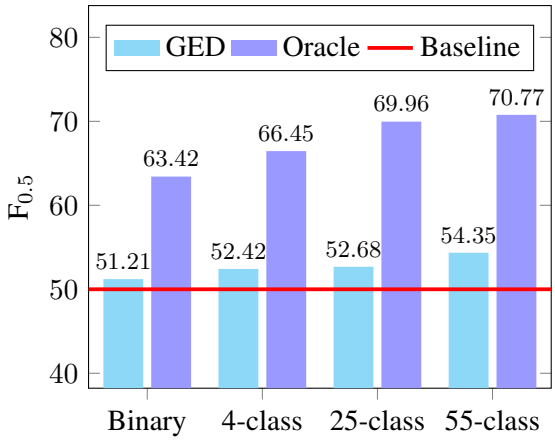

Figure 2: ERRANT span-level correction results on BEA-dev using GED predictions from different models for re-ranking.

It is finally worth mentioning that we do not use any other features in our approach – not even the original scores output by the GEC system – and so our simple re-ranking method can also be applied to any number of hypotheses from multiple systems.

Results We perform re-ranking using both GED predictions[8] and oracle labels as before. We report the results on BEA-dev using the binary, 4-class, 25-class and 55-class GED labels. As can be seen in Figure 2, using GED for re-ranking GEC output improves the results consistently and significantly.[9] Interestingly, there is a gradual increase in performance as the granularity of the GED labels also increases; the best re-ranking model uses 55-class GED (54.35 $F_{0.5}$). We suspect this is because the 55-class labels are the most discriminative compared to the other tagsets, even if they are potentially the noisiest. The performance boost using GED predictions is still far from what the model can achieve using oracle however. This suggests that having more accurate GED models is essential and that more fine-grained error types can be effectively incorporated into GEC.

Looking deeper into the performance of our best re-ranking GEC system which uses 55-class GED, we see that the model's $F_{0.5}$ score increases by $+9.97$ for missing and $+10.98$ for unnecessary word errors. The improvement is much smaller for replacement errors however ($+1.13$). This is not the case for oracle where the improvement for replacement errors is also significant ($+17.97$). Our error analysis reveals that the highest gain is achieved for `R:NOUN:INFL` ($+25.44$), followed by `U:VERB` ($+19.16$), `U:PUNCT` ($+16.63$), and `U:VERB:TENSE` ($+15.84$).[10]

5.5 Final GEC results

In our final experiment, we combine both methods and apply the 55-class GED re-ranking strategy with our best multi-encoder GEC model which uses 4-class GED in both the constrained and unconstrained settings. We also evaluate our models on two other GEC benchmarks: BEA-test and CoNLL-2014. Results are presented in Table 5, where further performance gains are observed, suggesting that these two methods are complementary.

Comparison with other systems In terms of the constrained setting, which only uses public data released in the BEA-2019 shared task, our system outperforms the only other comparable system by a large margin. Specifically, we outperform Raheja and Alikaniotis (2020) by $+9.4$ $F_{0.5}$ on BEA-test and $+7.3$ $F_{0.5}$ on CoNLL-2014, with the largest gains coming from re-ranking.

In terms of the unconstrained setting, which includes systems trained on additional private and/or artificial data, our system outperforms all other previous work that combines GED and GEC, and furthermore achieves a new single-model NMT-based state of the art on BEA-test. Our closest NMT-based competitor meanwhile is Stahlberg and Kumar (2021), who holds the current record on CoNLL-2014 (66.6 $F_{0.5}$). Although Omelianchuk et al. (2020) score higher than our approach on both test sets, we note that their sequence-tagging approach additionally relies on a carefully curated set of 5000 language-specific edit tags. Ultimately, we believe we have demonstrated the value of incorporating multi-class GED into GEC and also the effectiveness of our proposed approaches.

Error type performance We also perform a detailed error analysis to better understand the performance of our final GEC system.[11] The largest improvement over the baseline is observed in `U:CONJ` ($+64.52$), followed by `U:NOUN:POSS` ($+41.21$), `R:NOUN:INFL` ($+21.57$), `R:VERB:INFL` ($+20.00$), `M:PUNCT` ($+16.89$), `M:VERB` ($+15.12$), `U:PUNCT`

[8] The GED systems for re-ranking experiments are fine-tuned on in-domain FCE + W&I.

[9] We perform two-tailed paired T-tests, where $p < 0.001$.

[10] Throughout our error analysis, we note that type-specific results may not be truly representative as some error types only account for a small fraction of the test data. See Appendix D.

[11] More details in Appendix E.

	System	BEA-test			CoNLL-2014		
		P	R	$F_{0.5}$	P	R	$F_{0.5}$
constrained	**NMT-based systems**						
	Raheja and Alikaniotis (2020)	53.8	36.5	49.1	64.7	22.6	47.1
	NMT-based systems using GED						
	Kaneko et al. (2020)	58.1	44.8	**54.8**	63.6	33.0	**53.6**
	Our work						
	Baseline	49.0	41.9	47.4	51.0	26.8	43.2
	+ Multi-encoder GEC	54.0	44.6	51.8	53.8	31.3	47.0
	+ GED re-ranking	60.8	50.8	**58.5**	60.4	39.0	**54.4**
unconstrained	**NMT-based systems**						
	Ji et al. (2017)†	-	-	-	-	-	45.2
	Ge et al. (2018)†‡	-	-	-	61.2	37.9	54.5
	Kiyono et al. (2019)•	65.5	59.4	64.2	67.9	44.1	61.3
	Lichtarge et al. (2020)△▲	67.6	62.5	66.5	69.4	43.9	62.1
	Wan et al. (2020)°	66.9	60.6	65.5	69.5	47.3	63.5
	Stahlberg and Kumar (2021)△▲□	72.1	64.4	**70.4**	72.8	49.5	**66.6**
	Yuan and Bryant (2021)†	-	-	-	74.3	39.0	62.9
	NMT-based systems using GED						
	Zhao et al. (2019)♦	-	-	-	67.7	40.6	59.8
	Yuan et al. (2019)▲♦	70.5	55.1	66.8	-	-	-
	Kaneko et al. (2020)•	67.1	60.1	65.6	69.2	45.6	62.6
	Chen et al. (2020)▲•■	70.4	55.9	66.9	72.6	37.2	61.0
	Wang et al. (2020)▲	-	-	-	65.0	33.5	54.6
	Our work†						
	Baseline	70.0	50.9	65.1	72.6	34.4	59.4
	+ Multi-encoder GEC	70.8	57.2	67.6	73.8	39.3	62.7
	+ GED re-ranking	73.3	61.5	**70.6**	71.3	44.3	**63.5**
	Sequence labelling systems						
	Omelianchuk et al. (2020)◊	79.2	53.9	**72.4**	77.5	40.1	**65.3**

Table 5: Comparison of recent single-model GEC systems evaluated using ERRANT on BEA-test and M^2 (Dahlmeier and Ng, 2012) on CoNLL-2014. Constrained systems are trained only on public BEA-2019 shared task data, while unconstrained systems are variously trained on private and/or artificial data, including: †CLC (2M sentences), ‡non-public Lang-8 (3M), △Wikipedia revision histories (170M), ▲artificial Wikipedia (170M), °artificial Gigaword (16M), •artificial Gigaword (70M), ◊artificial one-billion-word (9M) with user-defined language-specific edit operations, ♦artificial one-billion-word (30M), □artificial Colossal Clean Crawled Corpus (200M), and ■artificial News Crawl.

(+14.34), and U:VERB:FORM (+13.89). After looking at the individual system performance in previous sections, we believe the 4-class GED fine-tuning process contributes the most to the improvement for U:CONJ, U:NOUN:POSS, and R:VERB:INFL amongst others, while the 55-class GED re-ranking process improves performance for R:NOUN:INFL, U:PUNCT, and U:VERB:TENSE. Meanwhile, both steps contribute to improvements in M:PUNCT and U:VERB:FORM, which shows how different GED systems benefit different error types. Finally, although our GEC system improves $F_{0.5}$ for most error types, we note that a small subset are negatively affected; e.g. performance on M:PART drops from 48.39 to 33.33, and R:ADJ:FORM from 75.00 to 62.50.

6 Conclusion

We have shown that multi-class GED can be used to significantly improve GEC. First, we showed that fine-tuning a pre-trained Transformer-based language model can lead to significant improvements in binary GED. Specifically, we found that fine-tuning ELECTRA, which has a discriminative

pre-training objective that is conceptually similar to GED, produces a new state of the art on three benchmark datasets. We furthermore showed that our models are capable of multi-class detection, and obtain similar $F_{0.5}$ performance to binary GED.

Next, we employ a multi-encoder GEC model and presented two methods of integrating GED predictions into GEC systems: firstly during GEC fine-tuning and secondly as a post-processing re-ranking step. Results show that both methods, when applied independently, significantly improve over a strong NMT-based GEC baseline. When applied together, we find the methods complement each other, yielding further performance gains. Our best single-model GEC system outperforms all previous systems that combine GED and GEC on both test sets, and all other single-model NMT-based systems on BEA-test.

Our results ultimately demonstrate the advantages of integrating multi-class detection into correction. In particular, different multi-class GED systems benefit GEC in different ways, and we find that the 4-class GED model leads to the best performance in fine-tuning the GEC system, but re-ranking using the 55-class GED model produces the best GEC performance overall. Finally, oracle experiments reveal that our proposed GEC systems are very effective at incorporating new GED information, but that there are still significant gains to be made in terms of more accurate GED systems.

Acknowledgements

We would like to thank Ted Briscoe, Paula Buttery and the anonymous reviewers for their valuable comments and suggestions. This paper reports on research supported by Cambridge Assessment, University of Cambridge. This work was performed using resources provided by the Cambridge Service for Data Driven Discovery operated by the University of Cambridge Research Computing Service, 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. We acknowledge NVIDIA for an Academic Hardware Grant.

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4260–4270, Hong Kong, China. Association for Computational Linguistics.

Samuel Bell, Helen Yannakoudakis, and Marek Rei. 2019. Context is key: Grammatical error detection with contextual word representations. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 103–115, Florence, Italy. Association for Computational Linguistics.

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

Mengyun Chen, Tao Ge, Xingxing Zhang, Furu Wei, and Ming Zhou. 2020. Improving the efficiency of grammatical error correction with erroneous span detection and correction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7162–7169, Online. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. Neural quality estimation of grammatical error correction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2528–2539, Brussels, Belgium. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *International Conference on Learning Representations*.

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.

Rachele De Felice and Stephen G. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 169–176, Manchester, UK. Coling 2008 Organizing Committee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24, Baltimore, Maryland. Association for Computational Linguistics.

Michael Gamon. 2011. High-order sequence modeling for language learner error detection. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 180–189, Portland, Oregon. Association for Computational Linguistics.

Tao Ge, Furu Wei, and Ming Zhou. 2018. 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.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2004. Detecting errors in English article usage with a maximum entropy classifier trained on a large, diverse corpus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762, Vancouver, Canada. Association for Computational Linguistics.

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, Shubha Guha, and Kenneth Heafield. 2018. 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.

Masahiro Kaneko and Mamoru Komachi. 2019. Multi-Head Multi-Layer Attention to Deep Language Representations for Grammatical Error Detection. *Computing Research Repository*, arXiv:1904.07334.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254, Online. 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 P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA.

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

Jared Lichtarge, Chris Alberti, and Shankar Kumar. 2020. Data weighted training strategies for grammatical error correction. *Transactions of the Association for Computational Linguistics*, 8:634–646.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *Computing Research Repository*, arXiv:1907.11692.

Zhenghao Liu, Xiaoyuan Yi, Maosong Sun, Liner Yang, and Tat-Seng Chua. 2021. Neural quality estimation with multiple hypotheses for grammatical error correction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5441–5452, Online. Association for Computational Linguistics.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

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

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*, pages 572–581.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

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

Vipul Raheja and Dimitris Alikaniotis. 2020. Adversarial Grammatical Error Correction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3075–3087, Online. Association for Computational Linguistics.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130, Vancouver, Canada. 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.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1191, Berlin, Germany. Association for Computational Linguistics.

Marek Rei and Helen Yannakoudakis. 2017. Auxiliary objectives for neural error detection models. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 33–43, Copenhagen, Denmark. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2011. Algorithm selection and model adaptation for ESL correction tasks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 924–933, Portland, Oregon, USA. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online. Association for Computational Linguistics.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202, Jeju Island, Korea. Association for Computational Linguistics.

Joel R. Tetreault and Martin Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 865–872, Manchester, UK. Coling 2008 Organizing Committee.

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

Zhaohong Wan, Xiaojun Wan, and Wenguang Wang. 2020. Improving grammatical error correction with data augmentation by editing latent representation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2202–2212, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Bo Wang, Kaoru Hirota, Chang Liu, Yaping Dai, , and Zhiyang Jia. 2020. An Approach to NMT Re-Ranking Using Sequence-Labeling for Grammatical Error Correction. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 24(4):557–567.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.

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.

Helen Yannakoudakis, Marek Rei, Øistein E. Andersen, and Zheng Yuan. 2017. Neural sequence-labelling models for grammatical error correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2795–2806, Copenhagen, Denmark. Association for Computational Linguistics.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

Zheng Yuan and Christopher Bryant. 2021. Document-level grammatical error correction. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 75–84, Online. Association for Computational Linguistics.

Zheng Yuan, Felix Stahlberg, Marek Rei, Bill Byrne, and Helen Yannakoudakis. 2019. Neural and FST-based approaches to grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 228–239, Florence, Italy. Association for Computational Linguistics.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 533–542, Brussels, Belgium. 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.

A Corpus statistics

Corpus	# Sentences
FCE-train	28,350
FCE-dev	2,191
FCE-test	2,695
W&I-train	34,308
BEA-dev	4,384
BEA-test	4,477
NUCLE	57,151
Lang-8	574,281
CLC	1,961,065

Table 6: Basic corpus statistics. Lang-8 only includes sentences that contain errors.

B Binary and multi-class GED performance

Training data	Mode	BEA-dev $F_{0.5}$		FCE-test $F_{0.5}$	
		binarised	macro	binarised	macro
CLC	binary	**67.18**	81.45	75.60	85.39
	4-class	66.86	67.86	**76.06**	72.89
	25-class	67.02	51.99	74.53	58.08
	55-class	67.01	47.47	74.94	47.54
Lang-8	binary	60.37	77.70	62.96	78.33
	4-class	**61.99**	63.39	**66.11**	64.00
	25-class	59.68	44.42	65.58	42.81
	55-class	60.40	37.84	64.64	37.52

Table 7: Binary and multi-class error detection performance of the ELECTRA GED model trained on the CLC and Lang-8. The highest binarised $F_{0.5}$ scores are in bold.

C Type-specific error analysis: 4-class multi-encoder GEC

Type	Baseline			4-class GED			4-class Oracle		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
M	54.14	30.26	46.76	54.14	50.72	53.42	71.72	77.50	72.80
R	59.68	32.38	51.07	61.75	33.68	52.93	64.66	58.29	63.28
U	64.69	26.57	50.27	60.59	29.01	49.76	78.83	81.26	79.30

Table 8: Precision, recall and $F_{0.5}$ for missing, replacement and unnecessary errors for baseline and our multi-encoder GEC systems fine-tuned with 4-class GED predictions and oracle on BEA-dev.

D Type-specific error analysis: 55-class re-ranking GEC

Type	Baseline			55-class GED			55-class Oracle		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
M	54.14	30.26	46.76	62.37	41.66	56.73	80.00	53.76	72.88
R	59.68	32.38	51.07	57.54	38.07	52.20	74.07	54.29	69.04
U	64.69	26.57	50.27	68.36	43.26	61.25	84.65	55.20	76.49

Table 9: Precision, recall and $F_{0.5}$ for missing, replacement and unnecessary errors for baseline and our re-ranking GEC systems using 55-class GED predictions and oracle on BEA-dev.

Type	Baseline $F_{0.5}$	55-class GED $F_{0.5}$	Diff. $F_{0.5}$	% of errors
R:NOUN:INFL	66.67	92.11	25.44	0.13
U:VERB	19.80	38.96	19.16	0.39
U:PUNCT	35.53	52.16	16.63	1.41
U:VERB:TENSE	43.86	59.70	15.84	0.56
M:PUNCT	46.20	60.72	14.52	14.61
U:VERB:FORM	41.67	55.56	13.89	0.16
U:PART	65.22	78.95	13.73	0.09
R:DET	38.94	51.66	12.72	2.12
U:PRON	57.55	70.18	12.63	0.63
R:CONTR	70.42	82.09	11.67	0.25
M:NOUN:POSS	70.09	80.81	10.72	0.36
M:PART	48.39	26.32	-22.07	0.15
R:PART	74.07	64.71	-9.36	0.56
U:ADJ	55.56	46.88	-8.68	0.16
R:VERB:INFL	80.00	71.43	-8.57	0.07
R:ADJ:FORM	75.00	68.18	-6.82	0.21

Table 10: Error type-specific performance before and after 55-class GED re-ranking on BEA-dev. We show results for a subset of error types that are mostly positively (top part) and negatively (bottom part) affected.

E Type-specific error analysis: our final GEC system

Type	Baseline $F_{0.5}$	GEC $F_{0.5}$	Diff. $F_{0.5}$	% of errors
U:CONJ	0	64.52	64.52	0.15
U:NOUN:POSS	35.71	76.92	41.21	0.13
R:NOUN:INFL	66.67	88.24	21.57	0.13
R:VERB:INFL	80.00	100.00	20.00	0.07
M:PUNCT	46.20	63.09	16.89	14.61
M:VERB	26.32	41.44	15.12	0.55
U:PUNCT	35.53	49.87	14.34	1.41
U:VERB:FORM	41.67	55.56	13.89	0.16
U:VERB:TENSE	43.86	57.69	13.83	0.56
U:PART	65.22	78.95	13.73	0.09
M:PART	48.39	33.33	-15.06	0.15
M:CONJ	14.85	0	-14.85	0.34
R:ADJ:FORM	75.00	62.50	-12.50	0.21
M:ADV	29.63	17.39	-12.24	0.36
M:ADJ	9.80	0	-9.80	0.20

Table 11: Error type-specific performance of the baseline and our final GEC system on BEA-dev. We show results for a subset of error types that are mostly positively (top part) and negatively (bottom part) affected.