Ensembling and Knowledge Distilling of Large Sequence Taggers for Grammatical Error Correction

Anonymous ACL submission

Abstract

In this paper, we investigate GEC sequence tagging architecture with focusing on ensembling of the recent cutting-edge Transformers' encoders in their Large configurations. We encourage ensembling models by majority votes on span-level edits because it's tolerant to the model architecture and vocabulary size. Our best ensemble achieves a new SOTA result, the $F_{0.5}$ score of 76.05 on BEA-2019 (test), even without pre-training on synthetic datasets. Also, we perform model distillation of a trained ensemble to generate new training synthetic datasets, "Troy-Blogs" and "Troy-1BW". Our best single sequence tagging model that is pre-trained on generated Troy- datasets in combination with publicly available synthetic PIE dataset achieves a near-SOTA[1] result of the $F_{0.5}$ score of 73.21 on BEA-2019 (test). The code, datasets, and trained models are publicly available[2].

1 Introduction

Grammatical Error Correction (GEC) task has a purpose to correct grammatical errors in natural texts. It includes correcting errors in spelling, punctuation, grammar, morphology, word choice, and others. Intelligent GEC system receives text containing mistakes and produces its corrected version. GEC task is complicated and challenging: the accuracy of edits, inference speed, and memory limitations are the topics of intensive research.

Currently, Machine Translation (MT) is the mainstream approach for GEC. In this setting, errorful sentences correspond to the source language, and error-free sentences correspond to the target language. Early GEC-MT methods leveraged phrase-based statistical machine translation (PBSMT) (Yuan and Felice, 2013). Then they rapidly evolved to sequence-to-sequence Neural Machine Translation (NMT) based on gated recurrent neural networks (Yuan and Briscoe, 2016) and recent powerful Transformer-based Seq2Seq models. They autoregressively capture full dependency among output tokens; however, it might be slow due to sequential decoding. (Grundkiewicz et al., 2019) leveraged Transformer model (Vaswani et al., 2017) which was pre-trained on synthetic GEC data and right-to-left re-ranking for ensemble. (Kaneko et al., 2020) adopted several strategies of BERT (Devlin et al., 2018) usage for GEC. Recently, (Rothe et al., 2021) built their system on top of T5 (Xue et al., 2021), a xxl version of T5 Transformer encoder-decoder model and reached new state-of-the-art results (11B parameters).

The sequence tagging approach that generates a sequence of text edit operations encoded by tags for errorful input text is becoming more common now. LaserTagger (Malmi et al., 2019) is a sequence tagging model that casts text generation as a text editing task. Corrected texts are reconstructed from the inputs using three main edit operations: keeping a token, deleting it, and adding a phrase before the token. LaserTagger combines a BERT encoder with an autoregressive Transformer decoder, which predicts edit operations. Parallel Iterative Edit (PIE) model (Awasthi et al., 2019) does parallel decoding, achieving quality that is competitive with the Seq2Seq models[3]. It predicts edits instead of tokens and iteratively refines predictions to capture dependencies. A similar approach is presented in (Omelianchuk et al., 2020). GECToR system uses various Transformers as an encoder, linear layers with softmax for tag prediction and error detection instead of a decoder. It also managed to achieve competitive results being potentially several times faster than Seq2Seq because of replacing autoregressive decoder with linear output layers.

[1] To the best of our knowledge, our best single model gives way only to much heavier T5 model (Rothe et al., 2021).

[2] http://github.com/to-appear-after-publication

[3] http://nlpprogress.com/english/grammatical_error_correction

Also, nowadays generation of synthetic data is becoming significant for most GEC models. Natural languages are rich, and their Grammars contain many rules and exceptions; therefore, professional linguists usually need to annotate high-quality corpora for further training of ML-based systems mostly in a supervised manner (Dahlmeier et al., 2013), (Bryant et al., 2019). At the same time, human annotation is expensive, so researchers are working on methods for augmentation of training data, synthetic data generation, and strategies for its efficient usage (Lichtarge et al., 2019), (Kiyono et al., 2019), (Stahlberg and Kumar, 2021). Most of the latest works use synthetic data to pre-train Transformer-based components of their models.

In this work, we are focusing on exploring sequence tagging models and their ensembles. Although most of our developments might eventually be applied to other languages, we work with English only in this study. Being a rich-resource language, English provides a highly competitive area for GEC task[3]. We leave dealing with other languages for future work.

2 Base System Overview

2.1 GECToR architecture

Our tagging models are inherited from the GECToR (Omelianchuk et al., 2020). To date, GECToR shows near-SOTA results on CoNLL-2014 and BEA-2019 benchmarks[3]. It is based on AllenNLP (Gardner et al., 2017) and HuggingFace's Transformers (Wolf et al., 2019) libraries, and its source code is freely available[4].

GECToR is a sequence tagging model which contains a Transformer-based encoder stacked with two output linear layers that are responsible for error detection and error correction. They are trained with a cross-entropy loss function to produce tags that encode token-level edits. Then iterative post-processing is performed. GECToR predicts the tag-encoded transformations for each token in the input sequence; it can then apply these transformations to get the modified output sequence.

Since some corrections in a sentence may depend on others, applying the GEC sequence-tagger only once may not be enough to correct the sentence entirely. Therefore, they use an iterative correction approach: it modifies the sentence by running the tagger on it again and repeat - up to four times (Fig. 1).

[4] https://github.com/grammarly/gector

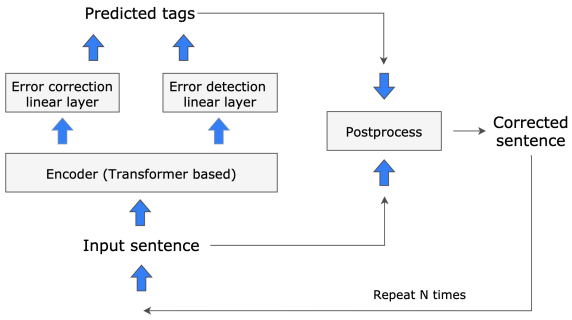

Figure 1: GECToR model: iterative pipeline.

2.2 Tag-encoded edit operations

The primary edit operations are encoded by the following tags: "KEEP" - leave the current token unchanged, "DELETE" - delete the current token, "APPEND TOKEN" - append the token "TOKEN" after the current token, "REPLACE TOKEN" - replace the current token with the token "TOKEN". Also, GECToR has special edit operations, such as transforming to uppercase or lowercase, transforming irregular verbs to their third forms, adding "s" word ending, etc. We refer to (Omelianchuk et al., 2020) for details of edit transforms.

2.3 Our contributions

We claim the following contributions:

1. We empirically investigate and improve the GECToR sequence tagging system (Omelianchuk et al., 2020) by upgrading Transformer encoders to Large configurations, leveraging advanced tokenizer, additional filtering of edits-free sentences, and increasing vocabulary size.

2. We show that ensembling of sequence taggers by majority votes on output edit spans provides better performance compared to ensembling by averaging of output tag probabilities while staying tolerant to models' architecture and vocabulary sizes.

3. We apply the knowledge distillation method to produce annotated data by the ensemble of sequence taggers. Being trained on the distilled data, single GEC tagging models show competitive performance.

4. We make the code, datasets, and trained models publicly available.

3 Datasets

3.1 Annotated data

For training single models and ensembles, we use parallel annotated data from Lang-8 Cor-

Dataset	Type	Part	# sent.	# tokens	% edits
Lang-8⋆	Ann	Train⋆	1.04M	11.86M	42%
NUCLE⋆	Ann	Train⋆	57k	1.16M	62%
FCE⋆	Ann	Train⋆	28k	455k	62%
W&I⋆†	Ann	Train⋆	34.3k	628.7k	67%
		Dev	3.4k	63.9k	69%
		Test†	3.5k	62.5k	N/A
LOCNESS†	Ann	Dev	1k	23.1k	52%
		Test†	1k	23.1k	N/A
1BW‡	Mon	N/A	115M	0.8B	N/A
Blogs‡	Mon	N/A	13.5M	171M	N/A
Troy-1BW	Dis	Train	1.2M	30.88M	100%
Troy-Blogs	Dis	Train	1.2M	21.49M	100%
PIE‡	Syn	Train	1.2M	30.1M	100%

Table 1: Description and statistics of datasets used in this work. Dataset types: (Ann)otated, (Syn)thetic, (Mon)olingual, and (Dis)tilled. ⋆Being combined these datasets form *Joint Train Dataset*. †BEA-2019 dev/test parts are concatenations of W&I and LOCNESS dev/test parts. ‡Only parts of original corpora from the cited sources are used in our work.

pus of Learner English (Lang-8)[5] (Tajiri et al., 2012), National University of Singapore Corpus of Learner English (NUCLE)[6] (Dahlmeier et al., 2013), First Certificate in English dataset (FCE)[7] (Yannakoudakis et al., 2011), and Write & Improve (W&I) Corpus (Bryant et al., 2019)[8]. Please, see Table 1 for details.

3.2 Monolingual data, distilled data

For knowledge distillation from the ensemble, we use parts of two monolingual datasets: One Billion Word Benchmark (1BW)[9] (Chelba et al., 2013) and The Blog Authorship Corpus (Blogs)[10] (Schler et al., 2005). Corresponding distilled datasets have prefixes "Troy-"; see more details about their generation in Section V.

3.3 Synthetic data

After knowledge distillation for the final training of the student model we also use parallel sentences with synthetically generated grammatical errors from the PIE dataset[11] (Awasthi et al., 2019).

[5] https://sites.google.com/site/naistlang8corpora

[6] https://www.comp.nus.edu.sg/~nlp/corpora.html

[7] https://ilexir.co.uk/datasets/index.html

[8] https://www.cl.cam.ac.uk/research/nl/bea2019st/data/wi+locness_v2.1.bea19.tar.gz

[9] http://statmt.org/wmt11/training-monolingual.tgz

[10] https://www.kaggle.com/rtatman/blog-authorship-corpus

[11] https://github.com/awasthiabhijeet/PIE/tree/master/errorify

3.4 Evaluation

We report $F_{0.5}$, $Precision$, and $Recall$ metrics computed by ERRANT scorer (Bryant et al., 2017) on dev, and test datasets from W&I + LOCNESS Corpus from BEA-2019 GEC Shared Task (Bryant et al., 2019).

4 Our System's Design

4.1 Tokenization

In the original GECToR code, the custom implementation[12] of the Byte-Pair Encoding (BPE) tokenizer (Sennrich et al., 2016) is used. It was chosen because out-from-the-box AllenNLP's tokenizer was too slow, and HuggingFace's Transformers' tokenizers did not provide BPE to words mapping. Our work is fully implemented with Transformers from the HuggingFace Transformers library. In particular, we moved to the recently released fast tokenizers from it. Now encoders have the same tokenizers for fine-tuning as they had for initial pre-training that leads to better quality after fine-tuning.

4.2 Initialization and training setup

Encoder is loaded with its default pretrained weights; the linear layers' weights are initialized with random numbers. Our models are trained by Adam optimizer (Kingma and Ba, 2015) with default hyperparameters. The loss function is a multi-class categorical entropy. The early stopping technique is used: stopping criteria is 3 epochs without improving the loss function on the dev set, which is random 2% from the same source as training data and is different for each stage.

4.3 Training stages

Model's training is performed during several stages. On Stage I, model is pretrained on synthetic datasets; this stage is optional. Then, on Stage II, we carry out warming training on the *Joint Train Dataset*, which contains Lang-8, NUCLE, FCE, W&I Train datasets (Table 1). Thus we perform coarse fine-tuning on a large amount of diverse GEC data. Datasets are used sequentially; no shuffling is made. Also, in order not to ruin out-of-the-box pretrained weights of the encoder, during the first two epochs, we train only linear layers (so-called "cold epochs"); later, we make all model's weights trainable.

[12] https://github.com/google/sentencepiece

On Stage III, we continue fine-tuning on the W&I Train dataset, which contains only the highest quality data. Another difference between Stages II and III is the share of edit-free sentences in the training data. We observed that too many sentences in training data without edits lead to reducing the appearance rate of the tagger and deteriorating the overall quality. Therefore, we filter out edit-free sentences from the Joint Train Dataset, which is used in Stage II. On Stage III, we fine-tune the model on the unfiltered version of the W&I Train dataset.

Training stage #	Base			Large		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Stage I.	N/A	N/A	N/A	N/A	N/A	N/A
Stage II.	50.12	34.04	45.79	52.11	37.34	48.29
Stage III.	53.77	**39.23**	50.06	54.85	**42.54**	51.85
Inf. tweaks	**62.49**	32.26	**52.63**	**65.76**	33.86	**55.33**

Table 2: Performance of our system with RoBERTa encoder after each training stage and inference tweaks on BEA-2019 (dev). Pre-training on synthetic data (Stage I) as was done in (Omelianchuk et al., 2020) is not performed.

The final stage is "inference tweaks" (Omelianchuk et al., 2020) for balancing between the model's precision and recall. It is performed by introducing additional hyperparameters: additional confidence (AC) to the probability for the KEEP tag and minimum error probability (MEP) for corrections tags. These hyperparameters are found through a random search on the BEA-2019 dev set.

4.4 Upgrading to Large encoders

In the GECToR paper (Omelianchuk et al., 2020), authors investigated encoders from ALBERT (Lan et al., 2020), BERT (Devlin et al., 2018), GPT-2 (Radford et al., 2018), RoBERTa (Liu et al., 2019), and XLNet (Yang et al., 2019) Transformers in their Base configurations. Most likely, Base configurations were chosen due to the better inference speed/quality ratio. They found that XLNet, RoBERTa, and BERT show the best quality.

We reproduce experiments for these encoders, but now we explore Large configurations as well. We additionally explore encoder from DeBERTa (He et al., 2020) (Table 3).

We observe that all models which are equipped with Large encoders have higher Precision, Recall, and $F_{0.5}$ values than those equipped with their Base versions. The price for it is 2.3 - 2.5 times slower inference for Large configurations (Table 4).

Encoder	Base			Large		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
BERT	57.21	29.93	48.39	61.18	31.26	51.35
DeBERTa	**64.22**	31.87	**53.38**	**66.35**	32.77	55.07
RoBERTa	62.49	**32.26**	52.63	65.76	33.86	**55.33**
XLNet	63.16	30.59	52.07	64.27	**35.17**	55.14

Table 3: Performance of our system on BEA-2019 (dev) for different encoders from pretrained Transformers in Base and Large configurations.

Encoder	Time, sec		# params	
	Base	Large	Base	Large
BERT	19.28	49.17	120M	350M
DeBERTa	23.75	58.32	150M	410M
RoBERTa	19.05	45.66	129M	360M
XLNet	30.46	71.19	120M	345M

Table 4: Inference times and models' sizes of our single tagging models. Inference time for NVIDIA Tesla P100 on BEA-2019 dev part, single models, batch size=128. Each value is an averaged time of 5 model inferences.

The single model with RoBERTa encoder shows the best performance among Large configurations, whereas DeBERTa slightly outperforms RoBERTa and is the best one among Base configurations. At the same time, RoBERTa remains the fastest one in both configurations.

4.5 Exploring tag vocabulary sizes

Most of the tag-encoded edits are token-specific, e.g., "APPEND Amelia", "REPLACE Brandon", and so on. Thus, the tag vocabulary size matters, and it should be a compromise between the covering of the natural language dictionary and the model's generalization abilities.

We create the tag vocabulary by taking the most frequent edit tags which were generated from the Joint Train dataset (Table 1). To find the optimal tag vocabulary sizes, we experiment with {5k, 10k} vocabulary sizes (Table 5).

Encoder	P	R	$F_{0.5}$
DeBERTa$^{(L)}_{5k}$	**66.35**	32.77	55.07
RoBERTa$^{(L)}_{5k}$	65.76	33.86	**55.33**
XLNet$^{(L)}_{5k}$	64.27	**35.17**	55.14
DeBERTa$^{(L)}_{10k}$	**65.46**	34.59	55.55
RoBERTa$^{(L)}_{10k}$	64.72	**36.04**	**55.83**
XLNet$^{(L)}_{10k}$	64.12	34.02	54.48

Table 5: Performance on BEA-2019 (dev) for varied tag vocabulary sizes and encoders in their (L)arge configurations. Subscripts encode the models' tag vocabulary sizes from the set {5k, 10k}.

We observe that increasing the vocabulary size to 10k for Large encoders may improve the quality,

as it happened for models with RoBERTa and DeBERTa. Nevertheless, also we see an example of quality deterioration for the model with XLNet.

5 Ensembling the GEC taggers

Ensembling is a proven quality boosting method for the models' sets that have diverse outputs. Most of the recent GEC solutions got their best results by ensembling single models (Stahlberg and Kumar, 2021), (Omelianchuk et al., 2020), (Awasthi et al., 2019). In this section we consider two ensembling methods for our GEC tagging models: averaging of output tag probabilities and majority votes on output edit spans (Fig. 2).

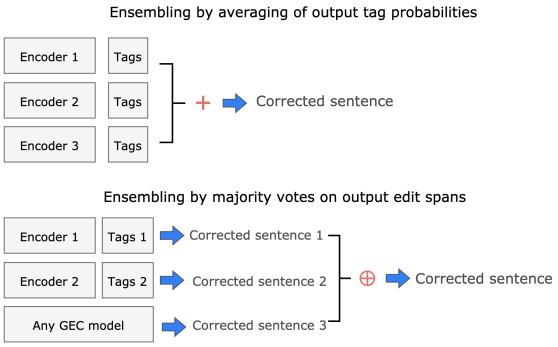

Figure 2: Ensembling by averaging of output tag probabilities (top) and ensembling by majority votes on output edit spans (bottom).

5.1 Exploring averaging of output tag probabilities (" + " operation)

First, we reproduce the ensembling approach from (Omelianchuk et al., 2020). We add DeBERTa and carry out experiments with varying Base and Large configurations of encoders (Table 6).

We observe that ensembling by averaging of output tag probabilities improves the quality of corrections; the more models we combine, the better results we obtain. More surprisingly, combining the same encoders' architectures in Base and Large configurations may provide slightly better results than we get for the Base and Large models separately, see RoBERTa$^{(B)}$ + RoBERTa$^{(L)}$ in Table 6.

Although the ensemble RoBERTa$^{(L)}$ + BERT$^{(L)}$ + DeBERTa$^{(L)}$ + XLNet$^{(L)}$ shows the best performance, we select ensemble the RoBERTa$^{(L)}$ + DeBERTa$^{(L)}$ + XLNet$^{(L)}$ for further experiments. It has higher Recall that makes it possible to trade Recall for Precision later during inference tweaks.

Ensemble	P	R	$F_{0.5}$
RoBERTa$^{(B)}$ + DeBERTa$^{(B)}$	53.44	**34.91**	48.31
RoBERTa$^{(B)}$ + XLNet$^{(B)}$	53.45	34.3	48.08
RoBERTa$^{(B)}$ + DeBERTa$^{(B)}$ + XLNet$^{(B)}$	54.78	34.87	49.17
RoBERTa$^{(B)}$ + BERT$^{(B)}$ + DeBERTa$^{(B)}$ + + XLNet$^{(B)}$	**56.34**	33.76	**49.69**
RoBERTa$^{(B)}$	50.12	34.04	45.79
RoBERTa$^{(L)}$	52.11	**37.34**	48.29
RoBERTa$^{(B)}$ + RoBERTa$^{(L)}$	**54.83**	35.93	**49.61**
RoBERTa$^{(L)}$ + DeBERTa$^{(L)}$	54.12	39.77	50.48
RoBERTa$^{(L)}$ + XLNet$^{(L)}$	53.83	38.65	49.91
RoBERTa$^{(L)}$ + BERT$^{(L)}$ + DeBERTa$^{(L)}$	**57.31**	37.41	51.8
RoBERTa$^{(L)}$ + DeBERTa$^{(L)}$ + XLNet$^{(L)}$	54.30	**39.95**	50.66
RoBERTa$^{(L)}$ + BERT$^{(L)}$ + DeBERTa$^{(L)}$ + + XLNet$^{(L)}$	56.97	38.52	**51.99**

Table 6: Comparison of ensembles by averaging of output tag probabilities after Stage II for (B)ase and (L)arge encoders, tag vocabulary size is 5k. Benchmark is BEA-2019 (dev).

5.2 Exploring majority votes on output edit spans ($\oplus$ operation)

This aggregation method combines single models' outputs on the post-processing step (Fig. 2). We take span-level edits and leave only those which have most of the votes from the ensemble. A similar approach is used in (Liang et al., 2020), where the authors combined sequence tagging and sequence-to-sequence models for the Chinese language. The advantage of this ensembling method is that we can combine the results of models with different output dimensions and even different architectures. In our work, it allows us to combine models with different tag vocabulary sizes. We leave ensembling with Seq2Seq GEC systems as a part of our future work.

First, we compare ensembling by averaging of output tag probabilities " + " and by majority votes on output edit spans $\oplus$ for the selected ensemble after training on Joint Train Dataset ("Stage II"), finetuning on W&I dataset ("Stage III") and optimization of hyperparameters ("inference tweaks") (Table 7). We observe that ensembles based on majority votes on output edit spans show better results because of better Precision. However, $F_{0.5}$ scores of both ensembling types are close to each other after inference tweaks.

To additionally improve the precision of ensembling by majority votes we introduce hyperparameter N_{min}, "majority quorum". Majority quorum N_{min} denotes *minumum number of votes for triggering the edit*, here $1 \leq N_{min} \leq N_{single\_models}$. Increasing N_{min} boosts the Precision by the cost of Recall because it filters out more edits where single

Stage	Ensemble	P	R	$F_{0.5}$
St. I	RoBERTa$^{(L)}$ + DeBERTa$^{(L)}$ + XLNet$^{(L)}$	N/A	N/A	N/A
St. I	RoBERTa$^{(L)}$ $\oplus$ DeBERTa$^{(L)}$ $\oplus$ XLNet$^{(L)}$	N/A	N/A	N/A
St. II	RoBERTa$^{(L)}$ + DeBERTa$^{(L)}$ + XLNet$^{(L)}$	54.3	**39.95**	50.66
St. II	RoBERTa$^{(L)}$ $\oplus$ DeBERTa$^{(L)}$ $\oplus$ XLNet$^{(L)}$	**56.74**	38.53	**51.84**
St. III	RoBERTa$^{(L)}$ + DeBERTa$^{(L)}$ + XLNet$^{(L)}$	58.08	**43.17**	54.33
St. III	RoBERTa$^{(L)}$ $\oplus$ DeBERTa$^{(L)}$ $\oplus$ XLNet$^{(L)}$	**60.58**	41.92	**55.63**
In.tw.	RoBERTa$^{(L)}$ + DeBERTa$^{(L)}$ + XLNet$^{(L)}$	68.45	**35.56**	57.76
In.tw.	RoBERTa$^{(L)}$ $\oplus$ DeBERTa$^{(L)}$ $\oplus$ XLNet$^{(L)}$	**69.67**	34.51	**57.88**

Table 7: Performance of selected ensemble for averaging of output tag probabilities ("+") and majority votes on output edit spans ("$\oplus$") ensembling types. Ensembles are not pre-trained on synthetic data (Stage I). Benchmark is BEA-2019 (dev).

models disagree (Table 8). Setting $N_{min} = 1$ is a poor strategy because we can't rely on the majority when resolving conflicting edits, so the resulting text might contain controversial and incoherent edits.

Increasing number of systems leads to higher quality, but requires adapting the N_{min} parameter (Table 8). Based on this limited analysis we observe that $N_{min} = N_{single\_models} - 1$ works the best. For our pool of models there is no gain over using more than 4 models, but we want to explore adding more diverse models based on Seq2Seq approach to such an ensemble in future works.

Next, since the majority votes on output edit spans is capable of combining any models, we test the ensemble of the best models that we already have trained (Table 9).

Finally, we evaluate our best ensemble DeBERTa$^{(L)}_{10k}$ $\oplus$ RoBERTa$^{(L)}_{10k}$ $\oplus$ XLNet$^{(L)}_{5k}$ on the BEA-2019 (test) dataset and achieve 76.05 of $F_{0.5}$ score. This is a significant improvement over $F_{0.5} = 73.70$ for the best ensemble from (Omelianchuk et al., 2020) and to the best of our knowledge **is a new state-of-the-art (SOTA) result for ensembles on BEA-2019 (test) benchmark**. It is worth noting that the solution is obtained without pre-training on synthetic data.

6 Knowledge distillation

Knowledge distillation is the method for transferring knowledge from a large model ("teacher") to a smaller one ("student") (Hinton et al., 2015), (Kim and Rush, 2016). It has strong practical applications because large models usually have expensive inference costs and are inconvenient for deployment.

In our case, the teacher model is an ensemble of trained sequence taggers, whereas the student model is a single sequence tagger. The ensemble receives errorful texts and generates their corrected versions. Later these input-output pairs of sentences are used for training single models. Of course, like any synthetic annotation method, knowledge distilled data contains a certain share of systematic errors that deteriorates the student model's quality.

6.1 Distilling the data.

In this work, we use two monolingual corpora to generate our distilled datasets: One Billion Words Benchmark ("1BW"), which mostly contains news, and The Blog Authorship Corpus ("Blogs"), which contains blog texts on various topics (Table 1). Being real-world natural texts, these datasets contain a certain share of grammatical errors, which are corrected by our system. For text pre-processing, we use the tokenizer from Spacy[13].

As a teacher, we use the ensemble of the sequence taggers containing Large encoders with 5k vocabulary: DeBERTa$^{(L)}_{5k}$ + RoBERTa$^{(L)}_{5k}$ + XLNet$^{(L)}_{5k}$ (Table 7). It corrects 5% of processed sentences in 1BW and 28% of sentences in Blogs datasets. Distilled versions of the datasets have the prefix "Troy-" in their names (Table 1). Considering our past experience, we fill our distilled datasets only with edited sentence pairs, and we limit their number to 1.2M. We also limit the synthetic PIE dataset from (Awasthi et al., 2019) to 1.2M sentence pairs for better comparability in the experiments. We leave exploring other ensembles in the role of a teacher model for future research.

6.2 Pre-training on synthetic and distilled datasets ("multi-stage training")

First, we reproduce the training scheme from (Omelianchuk et al., 2020) for a single model, RoBERTa$^{(L)}_{5k}$ where PIE synthetic data is used for pre-training (Stage I), then the model is trained on Joint Train Dataset (Stage II), after that it is trained on the high-quality W&I dataset (Stage III), and finally, a hyperparameter search of additional confidence probability and the minimum error probability is performed (Inf. tweaks). We observe that sequence tagger with RoBERTa-Large encoder shows slightly better performance than RoBERTa-Base from (Omelianchuk et al., 2020) where the

[13] https://spacy.io/

Ensemble	N_{single_models}	N_{min}	P	R	$F_{0.5}$
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k}$	3	1	44.49	**41.96**	43.96
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k}$	3	2	57.96	41.79	53.79
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k}$	3	3	**67.54**	30.99	**54.65**
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k} \oplus DeBERTa^{(L)}_{10k}$	4	1	40.21	41.68	40.50
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k} \oplus DeBERTa^{(L)}_{10k}$	4	2	55.02	**43.14**	52.15
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k} \oplus DeBERTa^{(L)}_{10k}$	4	3	64.48	37.49	**56.36**
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k} \oplus DeBERTa^{(L)}_{10k}$	4	4	**71.71**	27.89	54.57
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k} \oplus DeBERTa^{(L)}_{10k} \oplus XLNet^{(L)}_{10k}$	5	1	37.20	40.88	37.88
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k} \oplus DeBERTa^{(L)}_{10k} \oplus XLNet^{(L)}_{10k}$	5	2	51.77	**43.65**	49.92
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k} \oplus DeBERTa^{(L)}_{10k} \oplus XLNet^{(L)}_{10k}$	5	3	61.89	41.43	56.33
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k} \oplus DeBERTa^{(L)}_{10k} \oplus XLNet^{(L)}_{10k}$	5	4	56.43	34.43	**56.43**
$RoBERTa^{(B)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k} \oplus DeBERTa^{(L)}_{10k} \oplus XLNet^{(L)}_{10k}$	5	5	**73.12**	26.00	53.67

Table 8: Exploring an impact of N_{min} ("majority quorum"), a minumum number of votes to trigger the edit in majority votes ensembling. Benchmark is BEA-2019 (dev).

Ensemble	P	R	$F_{0.5}$
$DeBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{5k} \oplus XLNet^{(L)}_{5k}$	69.67	34.51	57.88
$DeBERTa^{(L)}_{10k} \oplus RoBERTa^{(L)}_{10k} \oplus XLNet^{(L)}_{10k}$	70.13	34.23	57.97
$DeBERTa^{(L)}_{5k} \oplus RoBERTa^{(L)}_{10k} \oplus XLNet^{(L)}_{5k}$	**70.71**	33.78	58.02
$DeBERTa^{(L)}_{10k} \oplus RoBERTa^{(L)}_{10k} \oplus XLNet^{(L)}_{5k}$	70.32	**34.62**	**58.30**

Table 9: Performance of best single models ensembled by majority votes on output edit spans. Subscripts encode the models' tag vocabulary sizes from the set {5k, 10k}. Benchmark is BEA-2019 (dev).

last one had an 8x larger training dataset on Stage I (Fig. 3).

Next, we replace the synthetic PIE dataset with our distilled datasets, Troy-1BW and Troy-Blogs. We observe that on Stage I, there is a difference with training on purely synthetic data that leads to the dramatic rise of Recall. However, when we start training on Stage II, a sharp deterioration in both Precision and Recall appears. It seems that the student model does not receive new information compared to Stage I. This is more noticeable for models trained on the Troy-Blogs dataset, which significantly drops Recall after training. At the same time, on Stage II, the $F_{0.5}$ is better for models pretrained on distilled Troy- datasets.

Finally, after training on Stage III and performing inference tweaks, single models pretrained on both datasets show very similar performance, but the model with $RoBERTa^{(L)}_{5k}$ trained on Troy-1BW was slightly better. **This single model reaches $F_{0.5} = 73.21$ on BEA-2019 (test), that significantly improves the results from (Omelianchuk et al., 2020) for single models** where they have $F_{0.5} = 71.5$ for $RoBERTa^{(B)}_{5k}$, and $F_{0.5} = 72.4$ for the $XLNet^{(B)}_{5k}$ encoders.

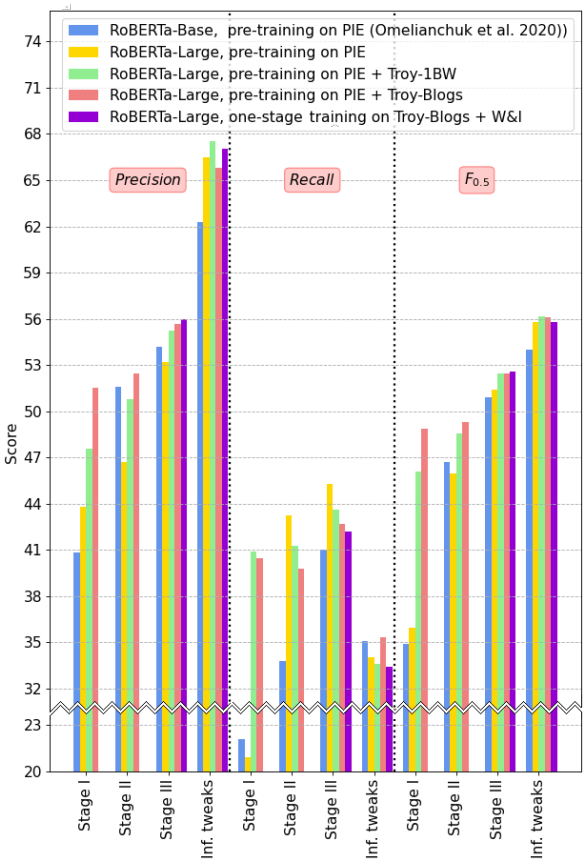

Figure 3: Pre-training of single tagging models on synthetic and distilled datasets, tags vocabulary size is 5k. Benchmark is BEA-2019 (dev).

6.3 One-stage training on distilled + annotated dataset

We observed that models which were pretrained on the Troy-Blogs dataset show good results on Stage I, but loose their advantage after training on Stage II. Thus, we trained one more model with $RoBERTa^{(L)}_{5k}$ encoder.

System	P	R	$F_{0.5}$
Single models			
(Kiyono et al., 2019)	65.5	59.4	64.2
(Omelianchuk et al., 2020)	79.2	53.9	72.4
(Kaneko et al., 2020)	67.1	60.1	65.6
(Stahlberg and Kumar, 2021)	72.1	**64.4**	70.4
(Rothe et al., 2021)	N/A	N/A	**75.88**
RoBERTa$^{(L)}_{5k}$, multi-stage training (this work)	**80.70**	53.39	73.21
RoBERTa$^{(L)}_{5k}$, one-stage training (this work)	80.55	52.27	72.69
Ensembles			
(Grundkiewicz et al., 2019)	72.3	60.1	69.5
(Kiyono et al., 2019)	74.7	56.7	70.2
(Omelianchuk et al., 2020)	79.4	57.2	73.7
(Kaneko et al., 2020)	72.3	61.4	69.8
(Stahlberg and Kumar, 2021)	77.7	**65.4**	74.9
DeBERTa$^{(L)}_{10k}$ $\oplus$ RoBERTa$^{(L)}_{10k}$ $\oplus$ XLNet$^{(L)}_{5k}$ (this work)	**84.44**	54.42	**76.05**

Table 10: Comparison of our best single tagging models and ensembles with related work on BEA-2019 (test).

We performed one-stage training where the Troy-Blogs dataset was concatenated with the most accurate W&I dataset that we usually use for Stage III. As a result, we got $F_{0.5} = 55.81$ on BEA-2019 (dev) and $F_{0.5} = 72.69$ on BEA-2019 (test) (Table 11). These results **are obtained much easier than our best single model: just one-stage training for out-of-the-box RoBERTa**, no pre-training on synthetic GEC data or multi-stage training.

7 Conclusions

Our best ensemble achieves a new SOTA result the $F_{0.5} = 76.05$ on BEA-2019 (test). Ensembling sequence taggers by majority votes on output edit spans provides better performance than averaging output tag probabilities while staying tolerant to models' architecture and vocabulary sizes. Single models in the ensemble were not pre-trained on synthetic GEC datasets that gives a room for improvement in future work.

We apply the knowledge distillation method to ensemble of sequence taggers for producing annotated Troy-Blogs and Troy-1BW datasets. After training on these datasets single GEC sequence tagging models show competitive results, $F_{0.5} = 73.21/72.69$ on BEA-2019 (test) for multi-stage/one-stage training. Replacing Base encoders in GECToR (Omelianchuk et al., 2020) with their Large configurations does improves the quality having up to x3 bigger size. However, in accuracy our best single model still gives way only to a much heavier T5 xxl model with 11B params (Rothe et al., 2021) having x30 less own size.

We make the code, datasets, and trained models publicly available[14].

[14] http://github.com/to-be-published

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 4259–4269.

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.

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)*, volume 1, pages 793–805.

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.

Ciprian Chelba, Tomás Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. 2018. 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.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer.

2017. Allennlp: A deep semantic natural language processing platform.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, Florence, Italy. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam (2014), a method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR), arXiv preprint arXiv*, volume 1412.

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.

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.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *ICLR 2020 : Eighth International Conference on Learning Representations*.

Deng Liang, Chen Zheng, Lei Guo, Xin Cui, Xiuzhang Xiong, Hengqiao Rong, and Jinpeng Dong. 2020. Bert enhanced neural machine translation and sequence tagging model for chinese grammatical error diagnosis. *Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 57–66.

Jared Lichtarge, Christopher 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.

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. *arXiv preprint arXiv:1907.11692*.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. 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 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem N. 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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A Simple Recipe for Multilingual Grammatical Error Correction. In *Proc. of ACL-IJCNLP*.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W. Pennebaker. 2005. Effects of age and gender on blogging. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 199–205.

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)*, volume 2, pages 198–202.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763.

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-Volume 1*, pages 180–189. 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.

Zheng Yuan and Mariano Felice. 2013. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 52–61, Sofia, Bulgaria. Association for Computational Linguistics.

A Appendix

System	P	R	$F_{0.5}$
Single models			
(Kiyono et al., 2019)	67.9	44.1	61.3
(Omelianchuk et al., 2020)	**77.5**	40.1	65.3
(Kaneko et al., 2020)	69.2	45.6	62.6
(Stahlberg and Kumar, 2021)	72.8	**49.5**	66.6
(Rothe et al., 2021)	N/A	N/A	**68.9**
RoBERTa$^{(L)}_{5k}$, multi-stage training (this work)	74.40	41.05	64.0
RoBERTa$^{(L)}_{5k}$, one-stage training (this work)	70.12	42.66	62.12
Ensembles			
(Grundkiewicz et al., 2019)	N/A	N/A	64.2
(Kiyono et al., 2019)	72.4	46.1	65.0
(Omelianchuk et al., 2020)	**78.2**	41.5	66.5
(Kaneko et al., 2020)	72.6	46.4	65.2
(Stahlberg and Kumar, 2021)	75.6	**49.3**	**68.3**
DeBERTa$^{(L)}_{10k}$ $\oplus$ RoBERTa$^{(L)}_{10k}$ $\oplus$ XLNet$^{(L)}_{5k}$ (this work)	76.1	41.6	65.3

Table 11: Comparison of our best single tagging models and ensembles with related work on CoNLL-14 (test).