BART & GECToR: An Unlikely Alliance for Grammatical Error Correction

Liao Tianchang
Niveditha Nerella
Samay Sagar
Wong Tun Hua Joshua
Wrik Karmakar
National University of Singapore

Abstract

In this project we propose a methodology that ensembles GECToR and BART models, aggregating their individual predictions to solve Grammatical Error Correction. We process the ensembling by applying (Tarnavskyi et al., 2022). We obtain improvements on SOTA on certain subtasks and analyse the composition of the features of these models that enable them to correct certain errors better than the peers. The proposed model achieves an $F_{0.5}$ of 61.30 on CoNLL-2014 (test) and $F_{0.5}$ of 69.03 on BEA-2019 (test). The experimented model files for BART, GECTOR, and GED can be accessed on the Sharepoint Folder, here.

1 Introduction

Grammatical Error Correction (GEC) is a foundational problem in Natural Language Processing that is defined as identifying and correcting different kinds of errors in text. These comprise spelling, punctuation, grammatical, and word choice errors.

Typically formulated as a sentence correction task, a potentially incorrect sentence is given as input and the corrected sentence is obtained as output. This, historically, aligned well with Neural Machine Translation approaches (Sennrich et al., 2016). The incorrect sentences form the source language whereas the correct sentences become the target language.

The NMT landscape radically improved after (Vaswani et al., 2017) introduced Transformer architectures. Predating those, LSTM models had been used to perform GEC (Makarenkov et al., 2019). However, with the landmark improvements obtained using transformers, this new methodology quickly drew improvements in GEC as well with emphasis on pretraining language models for the specific GEC task as well.

Omelianchuk et al., 2020 recognises that this formulation of the problem is married with a slow inference speed, obscurity of workings and a limiting reliance on large quantities of training data. Approaches to augment these methods with synthetic data confirm this further. Omelianchuk et al., 2020 then departs to a sequence based tagging mechanism that leverages the transformer architecture differently than in NMT.

Katsumata and Komachi, 2020, however, seeks to minimise the same high volume data reliance with a pretrained language model to minimise this cost. The pseudo-corpus approached to train Seq2seq models used requires generation of this synthetic data in large volumes as well as a time-consuming training processes. (Katsumata and Komachi, 2020) applies (Liu et al., 2019) to arrive at a competitive baseline for the GEC task whose pretraining made redundant any pseudodata, increasing efficiency.

Our approach, is to bridge these two methodologies together. The motivation behind this is to observe the synergies that exist as well as the differences between the two models. GECToR's methodology of training being derived from tagging also leads to a more explainable approach than BART which relies on excellent general pretraining - that led to great scores in more primary tasks such as text summarization.

The opportunity of ensembling led to (Tarnavskyi et al., 2022) which provided us a framework to compare other approaches to augment GECToR with and more importantly a stable and robust method to ensemble with GECToR, given the differences in how the prediction probability matrices are generated in either model.

2 Data

We use the training data from BEA 2019 Shared Task (Bryant et al., 2019) for GEC. We extracted 7 training datasets in the M2 format and split them to obtain source (`.src`) and target files (`.tgt`). We combine all the 7 source and target files to obtain - `train.src` and `train.tgt` which we use as the training data. We use the BEA-development dataset as the validation data while training the model. A similar M2-to-parallel processing is applied on the development dataset as well.

We use the BEA-2019 test dataset and CoNNL-2014 dataset (Ng et al., 2014) for obtaining the $F_{0.5}$ scores and evaluating the performance of our models.

3 BART: baseline model

Following Katsumata and Komachi, 2020, we trained the BART-based models by using `bart.large`. This model was proposed for the summarization task, which required some additional constraints in inference to ensure appropriate outputs; however, we did not impose any constraints because our task was GEC. We applied

	Dataset	
Training Data Combined size - [561,541] lines	FCE v2.1 Lang-8 Corpus of Learner English NUCLE (NUS Corpus of Learner English) W&I+LOCNESS v2.1	fce/m2/fce.train.gold.bea19.m2 fce/m2/fce.dev.gold.bea19.m2 lang8/lang8.train.auto.bea19.m2 nucle/bea2019/nucle.train.gold.bea19.m2 wi+locness/m2/A.train.gold.bea19.m2 wi+locness/m2/B.train.gold.bea19.m2 wi+locness/m2/C.train.gold.bea19.m2
Development data Size - [4384] lines	Bea-2019 Dev	wi+locness/m2/ABCN.dev.gold.bea19.m2
Test Data [4478] and [1313] lines respectively	Best-2019 Test CoNNL-2	wi+locness/test/ABCN.test.bea19.orig official-2014.combined.m2

Table 1: Summary of the datasets Used For BART and GECToR

byte pair encoding (BPE) (Sennrich et al., 2016) to the training and development data for the BART-based model by using the BPE model of Lewis et al., 2020.

3.1 Hyperparameter Tuning

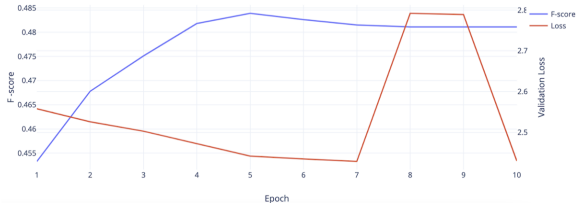

Figure 1: F-score and Loss for 10 epochs of BART

We trained the model for 10 epochs which generated 10 interim model checkpoints. The original BART was using the validation loss as an evaluation metric to pick the best model. However, we maximise the $F_{0.5}$ score on the validation data as a metric to pick the best model, instead of minimising the loss. Using this method, we picked `checkpoint5.pt` (model saved in the 5^{th} epoch) as our final BART model, as it had the highest training $F_{0.5}$ score of 0.4839.

Best Epoch based on minimum loss would be epoch 7, while the best epoch based on maximum F-score is epoch 5. This was our modified way of using the Baseline model. We used the M2 scorer (Dahlmeier and Ng, 2012) for CoNLL-14, and used the ERRANT scorer (Bryant et al., 2017) for the BEA-test. The scores we obtained were - **65.83** on **BEA-2019** and **62.83** on **CoNNL-2014**.

	Overall					
	TP	**FP**	**FN**	**P**	**R**	$F_{0.5}$
Span-level correction	2974	1345	2338	68.86	55.99	65.83
Span-level detection	3459	871	2089	79.98	62.35	75.63
Token-level detection	4262	554	2251	88.50	65.44	82.67

Table 2: Detailed Results of BART on BEA-2019

3.2 Grammatical Error Detection Model

In order to evaluate the performance of our baseline, we implemented a GED model which classifies an input sentence to be either grammatically correct or incorrect. It is at a high level, a binary classification task. We use 'The Corpus of Linguistic Acceptability' (CoLA) dataset (Warstadt et al., 2018) for single-sentence classification. It's a set of sentences labelled as grammatically correct or incorrect. We used the Hugging Face's PyTorch implementation for BERT with `BertForSequenceClassification` as the final layer in the model as it's a classification task. Our implementation is from Misra, 2020.

Figure 2: GED Results - BART corrects 1250/1313 sentences on CoNNL 2014 and 4218/4478 sentences in BEA-2019 which is classified as 'Correct' by GED

4 GECToR

The GECToR model we used for grammatical error correction applied sequence tagging for error detections and corrections, encoding via a pretrained RoBERTa (Liu et al., 2019) transformer and custom token-level transformations. Training and development data for our GECToR model used the same combined dataset - `train.src` and `train.tgt` as the BART model. We ran the training for 10 epochs and generated 10 interim model checkpoints. We took the best checkpoint which was based on the validation loss as an evaluation metric and used that as our trained model.

We used the M2 scorer (Dahlmeier and Ng, 2012) for CoNLL-14, and used the ERRANT scorer (Bryant et al., 2017) for the BEA-test. The scores we obtained were - **62.75** on **BEA-2019** and **58.50** on **CoNNL-2014**.

As an initial attempt into ensembling, we passed the test data through the BART model first, and followed that by running those results through the GECToR model. By doing so, we achieved **62.51** on the **BEA-test** and **61.32** on **CoNNL-2014**. We noticed that the score slightly dropped for BEA-2019 but increased by almost 2 points on CoNNL-2014. We realised that ensembling would be a reasonable method to try, but using a different approach instead of feeding the output of BART as an input to GECToR.

	Overall					
	TP	**FP**	**FN**	**P**	**R**	$F_{0.5}$
Span-level correction	3655	2080	2528	63.73	59.11	62.75
Span-level detection	4181	1563	2339	72.79	64.13	70.87
Token-level detection	5248	1180	2724	81.90	65.83	78.09

Table 3: Results of GECToR on BEA-2019

Precision :	0.6340
Recall :	0.4468
F_{05} = 0.5850	

Table 4: Results of GECToR on CoNNL-2014

	Overall					
	TP	**FP**	**FN**	**P**	**R**	$F_{0.5}$
Span-level correction	4341	2705	2197	61.61	66.40	62.51
Span-level detection	5042	2019	1876	71.41	72.88	71.70
Token-level detection	6328	1549	2099	80.34		79.23

Table 5: Results of BART2GECToR on BEA-2019

Precision :	0.6345
Recall :	0.5406
F_{05} :	0.6132

Table 6: Results of BART2GECToR on CoNNL-2014

5 Ensembling Methodology

We used the ensembling method implemented in (Tarnavskyi et al., 2022). The authors of that paper concluded that between averaging output tag possibilities and taking the majority votes on output edit spans, the latter performs better on their test set. As such, we replicated their ensembling script, which takes in the original unedited text, the edited output texts by the different models, and outputs the ensembled results through majority vote. We use the outputs of the BART and GECToR models, inputting them into the aforementioned program to ensemble the two, allowing us to overlay the strengths of each model with specific error corrections and detections to produce a model that showed significant improvements from either model. Using the majority-voting ensemble, the changes suggested by either model will have a lower probability of being accepted (as we used minimum count = 2), which will increase precision (lesser false positives) while sacrificing recall (higher false negatives), leading to an increase in the $F_{0.5}$ Score of **69.03** (as $F_{0.5}$ prioritises precision more than recall).

The ensembling function first tokenizes the original unedited text and the edited texts outputted by the different models. The tokenizer used here is from `en_core_web_sm` — a small English pipeline trained on written web text (blogs, news, comments), that includes vocabulary, syntax and entities — in the open-source NLP library `spaCy`. After tokenizing the inputs, the function uses the `SequenceMatcher` class in Python's builtin `difflib` library to compare each model's corrected text with the unedited text sequence by sequence, counting the correction made by the different models and including that correction in the final output if and only if both models have made the same correction for that sequence. By the mechanism mentioned above, we are able to ensemble the outputs of our two models to produce a new output.

6 Experiments: Comparative Analysis

6.1 Comparing the F-score by Operation Level

As evident from the figure, GECToR has a better performance score for the **'Missing'** operation level. However, BART scores better for the **'Replacement'** and **'Unnecessary'** operation levels. Thus, we decided to ensemble these models (BART and GECToR) in order to retain the individually better operation level corrections and thereby collectively improve the overall score. After ensembling these models, we saw improvement in the $F_{0.5}$ score for all the operation levels as the models complemented each other with respect to the operation level.

6.2 Comparing $F_{0.5}$ Score by Error Type Level

As evident from the figure, BART performs better than GECToR in almost every Error Type except **ORTH** and **NOUN:POSS**. For certain error types like **ADJ, ADV**, and **PART**, BART performs significantly better. Using an ensemble significantly improved the $F_{0.5}$ score for almost all error types, however in the case of **SPELL** and **VERB:INFL**, the error types with one of the highest scores for BART and GECToR, the ensemble led to a drastic fall in the score. One of the major reasons for **SPELL** might be the context required when detecting spelling corrections in a hard majority-vote ensemble ((**?**). However when it comes to **VERB:INFL**, no resource that would explain this discrepancy could be located.

7 Further Exploration

7.1 Alternative approaches to ensembling

Since we had already evaluated the models and they had good performance individually, we decided to do a *hard majority voting ensemble*. However, in the future, we can explore a *soft voting ensemble* by calculating the probabilities from each model and averaging them instead of directly performing a majority vote on the output sentences. We can also try out further extensions to voting ensembles such as a weighted average ensemble and stacked generalisation.

7.2 Differential analysis of Error Tags

The decrease in the performance of our ensemble for the **SPELL** and **VERB:INFL** tags is quite interesting as an avenue for exploration. Even though we found some explanation for **SPELL**, we still want to further analyse these two tags by modifying the training dataset and comparing the F-score for a *soft voting ensemble* that

	Overall					
	TP	**FP**	**FN**	**P**	**R**	$F_{0.5}$
Span-level correction	1822	315	2827	85.26	39.19	69.03
Span-level detection	1941	200	2831	90.66	40.67	72.77
Token-level detection	2236	118	3243	94.99	40.81	75.06

Table 7: Results of our ensemble on BEA-2019

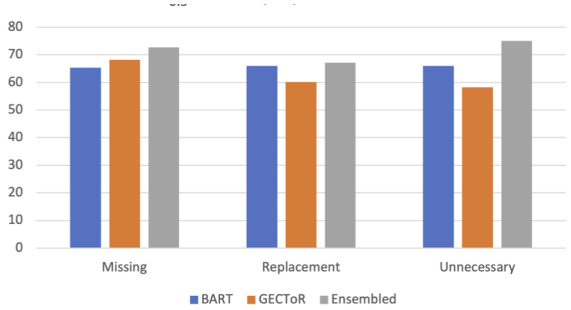

Figure 3: $F_{0.5}$ Score by Operation Level

has better context as compared to a *hard majority-vote ensemble.*

Furthermore, we use the model that minimises the loss on the validation data set for GECToR. However, in hindsight if we had maximised the $F_{0.5}$ score on the validation dataset like we had done for BART, the scores might have potentially improved as a result. A possible extension could be adding on more model outputs to our ensemble method and evaluate the results which we expect would be much higher than the baseline.

8 Conclusion

In this paper, we first implemented the baseline BART model and evaluated its output through the Grammatical Error Detection Model to gain further insights. We then explored GECToR as another possibility to compare with BART and found that our BART model performs better than GECToR, however, when it comes to **'Missing'** operation level corrections, the latter has better performance. Subsequently, we experimented with an ensemble of both BART and GECToR with a hypothesis that their operation level corrections complement each other and thus would lead to improved performance. Our hypothesis was correct with improvement in all three operation levels. However, we also found that such an ensemble led to a decrease in the F-score of certain errors that were performing much better for the models individually.

In the future, we plan to try out different approaches to ensembling including a probabilistic approach instead of hard majority voting to further improve our approach. Besides, we also want to explore the reason behind a decrease in the performance of **SPELL** and **VERB:INFL** in the ensemble.

9 Acknowledgement

The project was done under the CS4248 Natural Language Processing Module at the National University of Singapore (NUS). We also thank our Teaching Assistants, Muhammad Reza Qorib and Lin Ruixi, for their guidance and insightful comments.

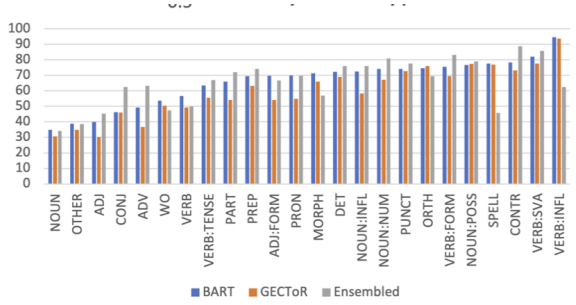

Figure 4: $F_{0.5}$ Score by Error Type

References

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.

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.

Satoru Katsumata and Mamoru Komachi. 2020. Stronger baselines for grammatical error correction using a pretrained encoder-decoder model. In *Proceedings of AACL-IJCNLP 2020*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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. *CoRR*, abs/1907.11692.

Victor Makarenkov, Lior Rokach, and Bracha Shapira. 2019. Choosing the right word: Using bidirectional LSTM tagger for writing support systems. *CoRR*, abs/1901.02490.

Sayak Misra. 2020. Checking grammar with bert and ulmfit. - towards data science. *Towards Data Science*.

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.

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 â†' Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Maksym Tarnavskyi, Artem Chernodub, and Kostiantyn Omelianchuk. 2022. Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. In *Accepted for publication at 60th Annual Meeting of the Association for Computational Linguistics (ACL 2022)*, Dublin, Ireland.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.