Improved Grammatical Error Correction through Neural Networks and Genetic algorithms

Niharika Shrivastava, Aditya Ganesh Kumar, Joey Li, Adi Kamaraj and **Lee Rong Jieh**
Department of Computer Science, National University of Singapore

Abstract

Grammatical Error Correction (GEC) is the task of correcting various types of errors in written English texts including spelling errors, punctuation errors, grammar errors, etc. In this paper, we build on top of the current SOTA ESC (Qorib et al., 2022) by exploring several independent methods to advance it's performance. Specifically, we modify the internal architecture of ESC to a simple CNN along with hyperparameter tuning. We also implement a Genetic Algorithm (GA) in combination with the ESC model. Additionally, we do a simple ensemble of ESC with the highest baseline and record our observations. Our CNN method successfully increases the F0.5 score from the highest base GEC system by 4.5 points on the CoNLL-2014 test set (Ng et al., 2014). Furthermore, our method outperforms the SOTA by 0.3 points on the CoNLL-2014 test set with and without the alternative annotation. We also show that our methods give higher precision by 1 point with better corrections.

1 Introduction

Grammatical error correction (GEC) is typically formulated as a sentence correction task that takes in potentially erroneous sentences as input and is expected to output corrected versions (Qorib et al., 2022). Most GEC systems implement a variation of Transformers (Bryant et al., 2017) along with a hybrid setup of Language Models. However, the current SOTA proposes an easy system combination of existing GEC models to predict better error corrections. The core of this method is to formulate a simple logistic regression problem using the outputs of the best existing Transformer-based GEC systems and propose probabilistic edits.

Moreover, the popularity of GEC has increased in the past years due to the publicly available pretrained GEC models including: bidirectional and auto-regressive transformers (BART) pretrained by predicting the original sequence, given a masked and shuffled sentence; and bidirectional encoder representations from transformers (BERT) pretrained unsupervised for context provision and understanding of ambiguous language (Devlin et al., 2019). With the use of generic pretrained encoder-decoder models as bases for our ensemble we can eliminate the time consumption from individually training systems. In this paper, we present several approaches to advance the ESC method for GEC, wherein the modified ESC architecture outperforms all prior SOTA systems on the CoNLL-2014 shared tasks, and the others (GA+ESC, ensemble with BART) beat the highest baseline.

The contributions of this paper are as follows:

- We propose a modified ESC architecture that makes use of 1D Convolution layers to decide on a proposed edit. We also implement an ensemble of our modified ESC architecture with the highest baseline BART.
- We combine the base ESC model with several variations of the Genetic Algorithm to select the best possible corrections out of the generated population.
- We also try various LM concepts such as Min-Edit-Distance spelling correction, and pseudo erroneous data generation but discard them as they were not scalable approaches.

2 Related Work

2.1 Easy Combination Systems

(Qorib et al., 2022) proposes a combination of systems for GEC as a simple machine learning task of binary classification. It uses a combination of GECToR-XLNet, UEDIN-MS, Riken-Tohoku, GECToR-Roberta, and T5-Large.

2.2 Genetic Algorithm

The genetic algorithm was invented by John Holland as a search and optimization tool based on Darwin's principle "Survival of the fittest"(Holland).

At each step, the algorithm selects models to be the parents and produce new offspring for the next generation by making slight modifications to the parents. The modifications are usually carried out by Crossover and Mutation, which means combining the parents and making random changes to the offspring. Eventually, we would achieve better fitted models after generations of 'evolution'.

3 Approach

The baseline for this paper was established using the neural network configuration described in (Qorib et al., 2022), the details of which is mentioned in Listing 1.

```
class Model(nn.Module):
  def init(self, feature_length):
      self.linear =
      nn.Linear(feature_length,1)
  def forward(self, x):
      x = torch.sigmoid(self.linear(x))
      return x
```

Listing 1: Baseline Neural Network

On top of the baseline, we modified the underlying network architecture along with a series of hyper-parameter tuning. Note that each modification of the baseline are independent of each other.

3.1 Addition of 1D Convolutions

Since the network decides which proposed edit to finally accept, a complex architecture is not needed to learn the underlying text features. Replacing a fully-connected linear layer with 2 blocks of Conv1D gave sufficient context to an edit based on its local neighbours along with a significant decrease in trainable parameters. Moreover, Adam optimiser with 1e-4 learning rate was found to perform better. The actual parameters of the convolution and max pool were trialled with different settings. We found that increasing kernel size beyond 3 resulted in decreasing performance. Also, padding was set to "same" to prevent loss of information. Max pooling was meant to condense information across layers. Here, using kernel size of 5 and a stride of 2 gave us better performance while a size of 3 resulted in lower performance.

```
Conv1d[(3x3) kernel, 'same' padding]
ReLU
MaxPool1d[(5x5) kernel, stride 2]
Conv1d[(3x3) kernel, 'same' padding]
ReLU
MaxPool1d[(5x5) kernel, stride 2]
Linear[ceil(feature_length/4) - 3 -> 1]
```

Listing 2: Improved Neural Network Configuration

3.2 Ensemble of Modified ESC with BART

BART (Lewis et al., 2019) currently provides the highest baseline for GEC. It uses a series of Transformers to implement a combination of masked-token generation and next-sentence prediction along with fine-tuning on a text-summarization task. We feed the output of the Convolutional-ESC defined in section 3.1 to this pre-trained BART model. It was observed that BART adds extra determinants in the final output and generally underfits the data thereby lowering down the overall score.

3.3 Genetic Algorithm for ESC Variants

We use Genetic Algorithm to greedily select the best variant of the ESC model out of a generated pool of candidates. We use the inverse of the binary cross entropy loss of each model as the fitness value, a larger fitness value would mean that the prediction is closer to the target.

- For crossovers, we have used single point crossover, two point crossover, uniform crossover and scattered crossover.
- For mutations, we have used random mutation, swap mutation, inverse mutation and scramble mutation.
- For parent selection methods, we experimented with the following (S.N.Sivanandam and S.N.Deepa., 2008):

Method	**Description**
`Steady-state`	Selection and worst-element deletion.
`Random`	Randomly select models.
`Tournament`	Models with highest fitness value becomes parents.
`Roulette wheel`	Random selection with probability related to fitness value.
`Stochastic universal`	Random selection with evenly spaced intervals.
`Rank`	Random selection with tournament selection.

Table 1: Parent Selection Methods

Parameter	**Value**
`No. Generations`	1000
`No. Parents Mating`	5

Table 2: Parameters for Genetic Algorithm

3.4 Character Level Error Correction

Since GEC also deals with spelling error correction, we decided to generate synthetic training data to improve character-level correction. We proposed a combination of addition, deletion, swap, substitution of noise characters (called "edits") which would be injected in the training data to teach the model common spelling mistakes. However, we faced 2 major challenges which resulted in us discarding this approach:

1. **Deciding the underlying distribution of characters to choose for the edits:** Using characters in proximity to the current character in the keyboard is a logical choice.

2. **The frequency at which to inject the edits:** The distribution must be close to the distribution of real-life human errors.

4 Results

4.1 Datasets

Evaluation of our GEC system was carried out on the CoNLL-2014 and BEA-2019 shared tasks.

The CoNLL-2014 (Ng et al., 2014) task involves the detection and correction of grammatical errors of all errors types in an English essay written by a learner of English as a second language. It includes 28 different error types ranging from incorrect word order to redundancy and evaluation is completed on the corrected output essay against the gold-standard edits adopting the F0.5 evaluation metric which emphasizes precision twice as much as recall. This is because in applied grammar correction, it is important to propose highly accurate corrections in order to gain user acceptance.

The second evaluation set is the BEA-2019 (Bryant et al., 2019) shared task. A new annotated data-set, the Cambridge English Write and Improve (W and I) and LOCNESS corpus, which is designed to represent a wider range of English levels and abilities than previous corpora including that of the previous traditionally used CoNLL-2014 shared task in hopes to develop systems better at generalizing on unseen data. Models are evaluated using ERRANT F0.5, which allows a greater range of performance statistics to be analyzed.

4.2 Observations

4.2.1 Improved Networks Architecture

Using the best neural network configuration described in Listing 2, we were able to obtain a small improvement from the baseline. From Table 3 and 4, we observe a noticeable gain in precision and slight improvements in recall and F0.5 scores. Our experiments suggest that using more linear or convolution layers beyond the current configuration leads to decreased performance.

Additionally, It is also important to note that there were several instances where the ground truth provided was incorrect, whereas our predicted output was correct. E.g ('suppose' is incorrect):

- **Ground truth:** "When we are diagnosed with certain genetic diseases, are we suppose to disclose this result to our relatives ?"

- **Predicted:** "When we are diagnosed with a certain genetic disease , are we supposed to disclose this result to our relatives ?"

This also incorrectly hampers the score.

However, there was a decrease in performance for BEA Shared task. Also, an ensemble with BART performed lower than the SOTA.

4.2.2 Genetic Algorithm

The uniform crossover would have a slightly better performance in our context. In general, we found that tournament selection would have a better performance, while roulette wheel selection and stochastic universal selection did not perform too well in our context.

Metric	Baseline	CNN	Ensemble
Precision	81.98	82.43	69.63
Recall	43.36	43.45	52.01
F0.5	69.51	69.89	65.21

Table 3: CoNLL-2014 Restricted

Metric	Baseline	CNN	Ensemble
Precision	84.68	85.09	70.01
Recall	45.73	45.81	51.35
F0.5	72.28	72.63	66.25

Table 4: CoNLL-2014 Alternate Annotations

Metric	Baseline	CNN
Precision	86.6	61.16
Recall	60.9	72.58
F0.5	79.9	63.15

Table 5: BEA-2019

Selection	Crossover	F0.5
rws	single_point	0.6910
rws	two_points	0.6913
rws	uniform	0.6914
rws	scattered	0.6907
sss	single_point	0.6913
sss	two_points	0.6919
sss	uniform	0.6919
sss	scattered	0.6921
sus	single_point	0.6905
sus	two_points	0.6910
sus	uniform	0.6913
sus	scattered	0.6907
rank	single_point	0.6913
rank	two_points	0.6913
rank	uniform	0.6921
rank	scattered	0.6916
random	single_point	0.6907
random	two_points	0.6922
random	uniform	0.6921
random	scattered	0.6910
tournament	single_point	0.6919
tournament	two_points	0.6921
tournament	uniform	0.6921
tournament	scattered	0.6921

Table 6: CoNLL-2014: Different GA models using swap mutation

Selection	Crossover	Mutation	F0.5
sss	single_point	random	0.6905
sss	single_point	inversion	0.6905
sss	single_point	scramble	0.6905
sss	two_points	scramble	0.6905
sss	uniform	random	0.6905
sss	uniform	inversion	0.6905
sss	uniform	scramble	0.6905
sss	scattered	random	0.6905
sss	scattered	inversion	0.6905
sss	scattered	scramble	0.6905

Table 7: CoNLL-2014: Samples of other mutation types

The swap mutation has a much better performance than other mutation types in our context. As the swap mutation has an outstanding performance(other mutation types could not exceed a F0.5 score of 0.6905 as seen in Table 7), we would be using swap mutation for all of our genetic models. The results are shown in Table 6.

We have also tried to implement a genetic algorithm on a CNN model, which had F0.5 score of only 68.03.

5 Analysis and Conclusion

In this work, we present novel methods in solving the problem of GEC. From the results above, it can be seen that modifying the underlying architecture provides a slight improvement over the current SOTA. Moreover, using various permutations of the Genetic Algorithm provides significant improvement over the highest baseline. However, when ESC is extended with BART, model performance deteriorates due to the addition of unnecessary determinants. Therefore, it makes sense to perform an ensemble with a model such as U-Det that gives less weightage to extra determinants in the final output.

Our CNN method successfully increases the F0.5 score from the highest base GEC system by 4.5 points on the CoNLL-2014 test set. Furthermore, our method outperforms the SOTA by 0.3 points on the CoNLL-2014 test set with and without the alternative annotation. We also show that our methods give higher precision by 1 point with better corrections. We also show how some of our predicted outputs are indeed correct and the ground truth is erroneous, thereby lowering our overall score. However, the CNN architecture doesn't work well on the BEA-Shared Task. We believe it's because our model is grossly overfitting the CoNLL-2014 Shared task.

We also explored multiple concepts of a Language Model including character-level spelling error correction. However, due to the large amount of training data vocabulary and limited memory, it wasn't a scalable approach.

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*. 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)*. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North*.

J. H. Holland. Genetic algorithms. *Scientific American*, 267(1):66–73.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Hwee Ng, wu siew mei, Ted Briscoe, Christian Hadiwinoto, Raymond Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. pages 1–14.

Muhammad Qorib, Seung-Hoon Na, and Hwee Tou Ng. 2022. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1964–1974, Seattle, United States. Association for Computational Linguistics.

S.N.Sivanandam and S.N.Deepa. 2008. *Introduction to Genetic Algorithms*. Springer.