# Improving Sequence Tagging approach for Grammatical Error Correction task

Maksym Tarnavskyi[1] and Kostiantyn Omelianchuk[2]

[1] Ukrainian Catholic University, Faculty of Applied Sciences, Lviv, Ukraine
tarnavskyi@ucu.edu.ua
[2] Grammarly, Kyiv, Ukraine
kostiantyn.omelianchuk@grammarly.com

**Abstract.** Grammatical Error Correction (GEC) is an important Natural Language Processing (NLP) task. It deals with building systems that automatically correct errors in written text. The main goal is to develop a GEC system that receives a sentence with mistakes and outputs a corrected version. One of the existing approaches to solve this task is the iterative sequence tagging approach. The main idea is that the model takes erroneous sentences as input and predicts sequences of tags that need to be applied to these sentences to turn them into correct ones. The desired result of this work is an improved sequence tagging model, which achieves better results on commonly accepted benchmark datasets. In order to improve the model, we want to explore the following things: 1) to discover the impact of large transformers or other training schemes, 2) to explore data-weighted training strategies, 3) to explore ensemble distillation techniques for improving a single model, 4) to explore the possibility to extend tagging operations space, 5) to explore combining sequence tagging with the sequence-to-sequence approach.

**Keywords:** natural language processing · grammatical error correction · sequence tagging

## 1 Introduction and motivation

### 1.1 Problem observation

In today's world, literacy is highly valued. Competent writing requires knowledge of various language rules, exceptions, grammar constructions. Lack of this knowledge or inattention can be the cause of making errors. People might have an unpleasant impression when they notice mistakes in the text. Therefore it is necessary to check it. However, manual validation is time-consuming, requires attention and in-depth knowledge. That is why systems that can automatically correct errors are beneficial and in demand. Such systems can be applied in many scenarios, such as writing essays, papers, reports, and emails. They significantly speed up checking, allow visually and interactively correct mistakes, and at the same time, teach grammar[27]. Research on this topic is actively developing, and significant progress has already been achieved.

### 1.2 Motivation

Grammatical Error Correction (GEC) is an important Natural Language Processing (NLP) task. It deals with building systems that automatically correct errors in written text. Previously, proofreading required professional linguistic skills. With the advent and development of GEC systems, more and more people have the opportunity to check their text and write more competently. We selected this research topic as it has significant social value and demand. Our goal is to explore existing state-of-the-art approaches to GEC tasks, analyze their advantages and disadvantages, suggest possible improvements, and develop an open-source grammar error correction system.

In this work, we first make a brief overview of the GEC topic. We start with an explanation of how we selected the literature for review. Next, we formulate the problem statement for GEC. After that, we cite a list of major public datasets that are available for use. We consider the metrics used to assess GEC models' quality and standard evaluation benchmarks. Then we mention the approaches that have made a significant contribution to the development of the GEC field. Further, we regard the current state-of-the-art models and existing training strategies. This review allows us to substantiate our proposed ideas for improvement. We formulate research hypotheses, present the first outcomes, and what remains to be done.

## 2 Related Work

### 2.1 The method for literature search and selection

To select relevant and representative literature, we used the following iterative approach. Firstly, we focused our search on articles that review the GEC field [26][29]. They provided information about the commonly accepted benchmarks [3][23], which evaluate existing approaches.

Using Microsoft Academics[3], we found articles that referred to these GEC benchmarks. Articles were sorted by time (from recent to older). We selected ten articles, which describe diverse approaches that have high results on the benchmarks. These ten became the "seed" for further search using the Controlled Snowball Sampling tool[4] [9]. It generated a catalog of selected relevant publications using analysis of citation relationships, probabilistic topic modeling, and snowball sampling. The resulting catalog contained a set of articles covering all significant research and achievements in the GEC field. The initial seed and result catalog can be found on this repository[5] .

We selected the final list of articles from the catalog based on the model's score on the standard benchmarks, the availability of citations, the novelty, and contribution to the GEC systems development.

[3] https://academic.microsoft.com/
[4] https://github.com/gendobr/snowball
[5] https://github.com/MaksTarnavskyi/gec

### 2.2 The problem formulation

The main goal is to develop a GEC system that receives a sentence with mistakes and outputs a corrected version. This system should be able to cope with different types of errors. They can be morphological, lexical, punctuation, and others. Both the quality of the corrections and the performance speed of the system are important. We discuss more details about the models and their training and evaluation process in the following sections.

### 2.3 Datasets observation

A comprehensive overview of the GEC area was done in [29]. It starts from observation of the main public datasets used for supervised learning of GEC models. These datasets are NUCLE[7], Lang-8[27], FCE [31], JFLEG [22], WriteImprove+LOCNESS [3]. They consist of parallel pairs of erroneous and grammatically correct sentences. More details about them are given in Table 1 based on information from [29].

Table 1: Statistics and properties of public GEC datasets.

| Corpus | Component | Sents | Tokens | Chars per sent | Sents Changed | Error Type |
|---|---|---|---|---|---|---|
| NUCLE | - | 57k | 1.16M | 115 | 38% | minimal |
| FCE | Train | 28k | 455k | 74 | 62% | minimal |
| | Dev | 2.1k | 35k | | | |
| | Test | 2.7k | 42k | | | |
| Lang-8 | - | 1.04 M | 11.86 M | 56 | 42% | fluency |
| JFLEG | Dev | 754 | 14k | 94 | 86 | fluency |
| | Test | 747 | 13k | | | |
| W&I | Train | 34.3k | 628.7k | 60 | 67% | - |
| | Dev | 3.4k | 63.9k | 94 | 69% | |
| | Test | 3.5k | 62.5k | - | - | |
| LOCNESS | Dev | 1k | 23.1k | 123 | 52% | - |
| | Test | 1k | 23.1k | - | - | |

### 2.4 Evaluation

Tools M2Scorer [6] and ERRANT [2] are most often used to assess text correction quality. For instance, ERRANT can automatically extract edits from parallel original and corrected sentences, classify them, compare their overlap with ground truth edits. Any edit with the same span and correction in target and corrected sentences is a true positive (TP). In contrast, unmatched edits in the corrected and target sentences are false positives (FP) and false negatives (FN), respectively. Tools return calculated Precision, Recall, and $F_{0.5}$ (which gives more weight to the Precision of the corrections than to their Recall).

The commonly accepted shared tasks CoNLL-2014[23] and BEA-2019[3] are used to check and compare existing GEC models' quality. These benchmarks test the ability of models to cope with all types of errors (punctuation, spelling, and others), and the sentences for them were annotated by professional linguists.

### 2.5 Development of GEC models

The initial GEC approaches were hand-crafted rule-based models developed by professional linguists (similar to [21]). However, it was complicated to describe so many rules and exceptions used in the language which would not contradict each other.

The first data-driven approaches were based on Statistical Machine Translation (SMT). An example of application is described in [33]. This work describes phrase-based statistical machine translation (PBSMT) for the automatic correction of errors. This model produces conditional Bayesian probabilities between transitions from erroneous phrases to grammatically correct phrases. It doesn't yield high performance on CoNLL-2013 shared task [13], but reveals many problems that require careful attention when building SMT systems for error correction.

The approach based on Neural Machine Translation (NMT) is first described in the article [32]. It used the Encoder-Decoder method. The encoder encodes an erroneous sentence into a vector, and the decoder generates an already corrected sentence based on this vector. Both the encoder and the decoder were RNNs[5] composing of GRU or LSTM [11] units. The model achieved the new SOTA $F_{0.5}$ 39 on CoNLL-2014[23] at that time.

The advent of the Transformer architecture [28] has made a significant contribution to NLP development, and most modern GEC approaches use BERT-based [8] encoders and decoders.

### 2.6 Recent sequence-to-sequence models

The application of BERT as an encoder and decoder for the GEC task is investigated in [12]. Authors evaluated three methods: (a) initialize an Encoder-Decoder GEC model using pre-trained BERT as BERT-init [15]; (b) pass the output of pre-trained BERT into the Encoder-Decoder GEC model as additional features (BERTfuse) [34]; (c) Combine the best parts of (a) and (b) in their new method (c), when first fine-tune BERT with the GEC corpus and then use the output of the fine-tuned BERT model as additional features in the GEC model. Their approach (c) had better performance with result of $F_{0.5} = 65.2$ at CoNLL-2014 [23] and $F_{0.5} = 69.8$ at BEA-2019 [3].

Sequence-to-sequence NMT-based approaches give quite good results but also have limitations. Their learning requires a lot of training data, and they are relatively slow. Therefore, researchers are trying to find methods that would speed up the performance of the models. One of them is the approach proposed in [4]. It improves GEC's efficiency by dividing the task into two subtasks: Erroneous Span Detection (ESD) and Erroneous Span Correction (ESC). ESD

identifies grammatically incorrect text spans with an efficient sequence tagging model. Then, ESC leverages a sequence-to-sequence model to take the sentence with annotated erroneous spans as input and only outputs the corrected text for these spans. Experiments show that their approach performs comparably to conventional sequence-to-sequence models, having $F_{0.5} = 61.0$ on CoNLL-14[23] benchmark with less than 50% time cost for inference.

### 2.7 Recent sequence tagging models

Most of what output the sequence-to-sequence Encoder-Decoder models is almost the same sentence fed to the input. The Encoder-Decoder models autoregressively capture full dependency among output tokens but are slow due to sequential decoding. A much more straightforward task is to predict the edit tags that need to be applied to turn erroneous sentences into correct ones.

This approach was proposed in the article [20]. The authors introduced LaserTagger - 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. The model combines a BERT encoder with an autoregressive Transformer decoder, which predicts edit operations. LaserTagger had $F_{0.5} = 40.5$ on the BEA-19 [3] test. However, it gave great impetus to the development of sequence tagging models.

Another approach was proposed in the article [1]. Their Parallel Iterative Edit (PIE) model does parallel decoding, giving competitive accuracy with the Encoder-Decoder models. This is possible because model:

- Predict edits instead of tokens.
- Label sequences instead of generating sequences.
- Iteratively refine predictions to capture dependencies.
- Factorize logits over edits and their token argument to harness pre-trained language models like BERT.

This method achieves $F_{0.5} = 59.7$ on the CoNLL-14[23] task and is a significantly faster alternative for local sequence transduction.

A similar approach, which currently has state-of-the-art results, is proposed in the article [24]. The authors present a simple and efficient GEC sequence tagger using a Transformer as an encoder and a softmax layer for tag prediction as a decoder. In their experiments, encoders from XLNet[30] and RoBERTa[19] outperform three other cutting-edge Transformer encoders (ALBERT[16], BERT[8], and GPT-2[25]). The authors always used pre-trained transformers in their Base configurations. Their GEC system is pre-trained on synthetic data and then fine-tuned in two stages: first on errorful corpora and second on a combination of errorful and error-free parallel corpora. Also, they designed custom token-level transformations to map input tokens to target corrections. These transformations increase grammatical error correction coverage for limited output vocabulary size for the most common grammatical errors, such as *Spelling, Noun Number, Subject-Verb Agreement,* and *Verb Form* [32].

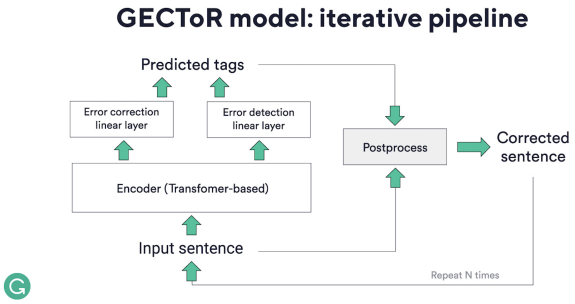


Fig. 1: GECToR model: iterative pipline.

Source: https://www.grammarly.com/blog/engineering/gec-tag-not-rewrite

Their best single-model, GECToR (XLNet) achieves $F_{0.5} = 65.3$ on CoNLL-2014[23] (test) and $F_{0.5} = 72.4$ on BEA-2019[3] (test). Best ensemble model, GECToR (BERT + RoBERTa + XLNet) where they simply average output probabilities from 3 single models achieves $F_{0.5} = 66.5$ on CoNLL-2014[23] and $F_{0.5} = 73.6$ on BEA-2019[3], correspondingly. Their inference speed is up to 10 times as fast as a Transformer-based sequence-to-sequence GEC system.

### 2.8 Data augmentation techniques and pre-training strategies

Encoder training requires as much quality data as possible. However, the most extensive set of publicly available parallel data (Lang-8[27]) in GEC has only one million sentence pairs. Therefore, many researchers are now actively exploring data augmentation methods and strategies for their use.

For instance, in the investigation [14], the authors proposed two approaches to error generation. In the first approach (a1), they set the probability distributions of different types of errors, such as deleting, repeating, substituting words in a sentence, and then directly added these errors to the correct sentences. In the second approach (a2), they trained the Encoder-Decoder Transformer model [28] in a reversed way, which received correct sentences as input and returned sentences with errors at the output. As a result, the GEC model, which trained on the data generated by approach (a1), had better results on the BEA-19 test [3].

Furthermore, they compared two strategies for model training on synthetic data. In the first experiment, they trained the model on the joined synthetic and real data. As a result, they got a slightly worse score than when they trained the model only on real data. In the second, they first pre-trained the model only on synthetic data and then trained the model only on real data. The second approach yielded significant improvements. This investigation demonstrated that training strategy is essential for outcome. Their final best pre-trained models achieved $F_{0.5} = 65.0$ on CoNLL-14 [23] and $F_{0.5} = 70.2$ on BEA-19 [3].

Another investigation is presented in the article [18]. The authors proposed the metric delta-log-perplexity (*Δppl*), defined as the difference in negative log-probability (log-perplexity) of an individual training example between two checkpoints in model training. The first checkpoint corresponds to model ($\theta^-$), which trained on a base dataset $D^-$, while the second checkpoint corresponds to the model ($\theta^+$) after further finetuning on a second target dataset $D^+$ (with trusted quality). *Δppl* between those models for a given example (composed of input, output pair $(i, o)$) should suggest which of the datasets the example is more similar to, from the successive models $\theta^-$ and $\theta^+$.

$$\Delta ppl(i, o; \theta^-, \theta^+) = log\ p(o|i; \theta^-) - log\ p(o|i; \theta^+)$$

When $D^+$ is selected to be 'higher quality' than $D^-$, then the *Δppl* scores of examples drawn from $D^-$ provide a heuristic for assessing their quality. Having these scores, the authors proposed a strategy for their use. First, they sorted the training examples according to scores. Since the model can forget more what it learned before and remember what it learned recently, the authors gave fewer quality examples at the beginning of training and high-quality - at the end. Thus, the model was able to study these more meaningful examples better and, as a result, achieved higher results on benchmarks. Also, they down-weighted the loss of low-scoring examples during training, which further improved the final result.

Using the weighting strategy, they increased $F_{0.5}$ for their single / ensemble models from 61.1 to 62.1 / from 65.3 to 66.8 on CoNLL-14[23] and from 66.1 to 66.5 / from 71.9 to 73.0 on BEA-19[3] tests.

To sum up, the articles in this section have shown that it is essential to choose not only the model's architecture but also its training strategy.

### 2.9 Analysis

The comparison of model's performance is presented in Table 2.

Table 2: The evaluation of models on GEC benchmarks

| | **CoNLL-2014.** | **BEA-2019** |
|---|---|---|
| **Model** | $F_{0.5}$ | $F_{0.5}$ |
| (Yuan et al., 2016) | 39 | - |
| (Kaneko et al., 2020) | 65.2 | 69.8 |
| (Chen et al., 2020) | 61.0 | - |
| (Malmi et al., 2019) | - | 40.5 |
| (Awasthi et al., 2019) | 59.7 | - |
| (Omelianchuk et al., 2020) - single | 65.3 | 72.4 |
| (Omelianchuk et al., 2020) - ensamble | 66.5 | 73.6 |
| (Kiyono et al., 2019) | 65.0 | 72.0 |
| (Lichtarge et al., 2020) - single | 62.1 | 66.5 |
| (Lichtarge et al., 2020) - ensemble | 66.8 | 73.0 |

## 3 Problem Setting and Approach to Solution

The task is to build a system that receives a sentence with errors and returns the corrected sentence. Our investigation focuses on the latest sequence tagging approach based on work [24]. It is an iterative model consisting of a Transformer-based encoder and a linear layer as a decoder. It takes erroneous sentences as input and predicts sequences of tags that need to be applied to these sentences to turn them into correct ones. Our goal is to explore methods that can improve its precision and recall metrics of error correction and the speed of inference.

To evaluate the experiments, we use the development part of the BEA-19 benchmark dataset. The metric for optimization is the $F_{0.5}$ score.

First of all, we want to compare the impact of different Transformer-based encoders and their configurations on the model's quality. Models using Roberta[19] and XLNet[30] encoders trained on the GEC corpus showed the best results in [24]. However, the authors used only the base configurations, while large transformers' impact remains unexplored. We want to test the hypothesis of whether increasing the configurations of transformers will lead to improved results. We also want to compare the usage of the latest transformers, such as DeBERTa[10], BART[17], and others.

After that, we want to test how the output tag vocabulary size affects accuracy and inference speed. In [24], the authors used 5000 edit tags for correction. However, the impact of vocabulary size on performance remains unexplored. We want to check if increasing the number of tags will improve the quality of the model.

The quantity and quality of training data and their use strategy are important for GEC[14][18]. That is why we want to discover the data weighting techniques for the selection of high-quality data. Since the delta perplexity heuristic cannot be used for the sequence tagging model, as in [18], we want to try our own method. We want to compare the similarity of the sentence embeddings obtained by the base model with the parameters $\theta^-$ and the model $\theta^+$ which is finetuned on the better quality dataset. We will select top K sentences, which will have the biggest/lowest similarity between embeddings, and try to add them into the target dataset and check performance after training on such an extended dataset.

It is worth noting that many studies report the final result obtained by a single model and by an ensemble of models. The ensemble's score is higher, while the performance is slower than for the single model. In our investigation, we also want to discover the distilling knowledge approach from the ensemble to the single model, improving its quality without slowing down its speed. We first train the ensemble in the usual way and use it to re-annotate low and moderate-quality data. After that, we want to train a single model on re-annotated data.

Another thing that can be explored is the possibility of extending tagging operations space. Adding new edit tags might improve corrections quality.

## 4 Early Results and Discussion

We are currently at the beginning of our research plan. However, we already have the first positive results.

We started with an experiment that tests the effect of large transformer configurations on the corrections' quality. We trained the GEC models, which used RoBERTa[19] and XLNet[30] encoders on the base and large configurations, and compared them with the results obtained in [24] for Stage 2.

To train the models, we used the joined datasets FCE[31], Lang-8[27], Nucle[7], and WI + Locness[3]. We split each dataset into training/validation parts in a ratio of 98/2, except for WI + Locness, for which we used existing partitions. For final validation, we used the BEA-19[3] development part. The size of the output tag vocabulary was 5000. The obtained results are shown in Table 3.

Table 3: A comparison of the results obtained using different encoders.

| Encoder | BEA-2019 (dev) | | |
|---|---|---|---|
| | *Precision* | *Recall* | $F_{0.5}$ |
| (Omelianchuk et al., 2020) | | | |
| ALBERT | 43.8 | 22.3 | 36.7 |
| BERT | 48.3 | 29.0 | 42.6 |
| GPT-2 | 44.5 | 5.0 | 17.2 |
| RoBERTa | 50.3 | 30.5 | 44.5 |
| XLNet | 47.1 | 34.2 | 43.8 |
| Our base transformers | | | |
| RoBERTa | 50.2 | 31.5 | 44.9 |
| XLNet | 48.2 | 34.1 | 44.5 |
| Our large transformers | | | |
| RoBERTa | 49.8 | 38.6 | 47.1 |
| XLNet | 48.8 | 39.4 | 46.6 |

As we can see, we were able to reproduce the experiment in [24] and improve the result using larger Transformer-based encoders. Models using large configurations of RoBERTa[19] and XLNet[30] performed significantly better than the corresponding models with base configurations. This result shows the truth of our first hypothesis. However, to fully confirm it, we need to train the models on all Stages 1-3, not only on Stage 2.

As for other hypotheses, we are still working on them, and we will discuss their results in future works.

## 5 Summary and Future Work

Grammatical Error Correction systems are beneficial and in demand. They allow automatic correction of various errors in written text. Despite significant recent progress in this area, the corrections quality and speed inference questions remain still open. This work provided a brief overview of the main existing GEC approaches, discussed their advantages and disadvantages, outlined research gaps, and proposed possible improvements. We are at the beginning of our research plan. However, we have already achieved the first positive results. Using large transformers, we got a better score of $F_{0.5}$ metric on the BEA-2019 dev part. However, to confirm this hypothesis, it is necessary to train the model at all stages and evaluate it on the BEA-19 test part. In the next steps, we plan to explore other modern Transformer-based encoders for the sequence tagging approach, investigate the impact of the output tag vocabulary size on quality and performance. We also plan to extend training data using data weighting techniques. Moreover, we want to explore the ensemble distillation methods for a single model. Finally, in the future, we plan to explore the possibility of extending the edit space of tags and combining sequence tagging methods with sequence-to-sequence methods. In our research, we adapt and apply the best-practice methods that have performed well for sequence-to-sequence models to sequence tagging models. The desired result of this work is an improved sequence tagging model, which achieves better results on commonly accepted benchmark datasets.

## References

1. Awasthi, A., Sarawagi, S., Goyal, R., Ghosh, S., Piratla, V.: 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). pp. 4259–4269 (2019)
2. Bryant, C., Felice, M., Briscoe, T.: 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). vol. 1, pp. 793–805 (2017)
3. Bryant, C., Felice, M., Øistein E. Andersen, Briscoe, T.: The bea-2019 shared task on grammatical error correction. In: Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications. pp. 52–75 (2019)
4. Chen, M., Ge, T., Zhang, X., Wei, F., Zhou, M.: 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). pp. 7162–7169 (2020)
5. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734 (2014)

6. Dahlmeier, D., Ng, H.T.: 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. pp. 568–572 (2012)
7. Dahlmeier, D., Ng, H.T., Wu, S.M.: 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. pp. 22–31 (2013)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.N.: 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). pp. 4171–4186 (2018)
9. Dobrovolskyi, H., Keberle, N.: On convergence of controlled snowball sampling for scientific abstracts collection. In: International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications. pp. 18–42 (2018)
10. He, P., Liu, X., Gao, J., Chen, W.: Deberta: Decoding-enhanced bert with disentangled attention. arXiv preprint arXiv:2006.03654 (2020)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
12. Kaneko, M., Mita, M., Kiyono, S., Suzuki, J., Inui, K.: 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. pp. 4248–4254 (2020)
13. hui Kao, T., wei Chang, Y., wen Chiu, H., Yen, T.H., Boisson, J., cheng Wu, J., Chang, J.S.: Conll-2013 shared task: Grammatical error correction nthu system description. In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task. pp. 20–25 (2013)
14. Kiyono, S., Suzuki, J., Mita, M., Mizumoto, T., Inui, K.: 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). pp. 1236–1242 (2019)
15. Lample, G., Conneau, A.: Cross-lingual language model pretraining. arXiv preprint arXiv:1901.07291 (2019)
16. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. In: ICLR 2020 : Eighth International Conference on Learning Representations (2020)
17. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: 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. pp. 7871–7880 (2020)
18. Lichtarge, J., Alberti, C., Kumar, S.: Data weighted training strategies for grammatical error correction. Trans. Assoc. Comput. Linguistics **8**, 634–646 (2020)
19. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
20. Malmi, E., Krause, S., Rothe, S., Mirylenka, D., Severyn, A.: 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). pp. 5053–5064 (2019)
21. Naber, D.: A rule-based style and grammar checker (01 2003)
22. Napoles, C., Sakaguchi, K., Tetreault, J.R.: Jfleg: A fluency corpus and benchmark for grammatical error correction. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. pp. 229–234 (2017)
23. Ng, H.T., Wu, S.M., Briscoe, T., Hadiwinoto, C., Susanto, R.H., Bryant, C.: The conll-2014 shared task on grammatical error correction. In: Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task. pp. 1–14 (2014)
24. Omelianchuk, K., Atrasevych, V., Chernodub, A.N., Skurzhanskyi, O.: Gector – grammatical error correction: Tag, not rewrite. In: Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications. pp. 163–170 (2020)
25. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2018), https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf
26. Rauf, S.A., Saeed, R., Khan, N.S., Habib, K., Gabrail, P., Aftab, F.: Automated grammatical error correction:a comprehensive review. NUST Journal of Engineering Sciences **10**(2), 72–85 (2018)
27. Tajiri, T., Komachi, M., Matsumoto, Y.: 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). vol. 2, pp. 198–202 (2012)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. vol. 30, pp. 5998–6008 (2017)
29. Wang, Y., Wang, Y., Liu, J., Liu, Z.: A comprehensive survey of grammar error correction. arXiv preprint arXiv:2005.06600 (2020)
30. Yang, Z., Dai, Z., Yang, Y., Carbonell, J.G., Salakhutdinov, R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. In: Advances in Neural Information Processing Systems. vol. 32, pp. 5753–5763 (2019)
31. Yannakoudakis, H., Briscoe, T., Medlock, B.: 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. pp. 180–189 (2011)
32. Yuan, Z., Briscoe, T.: 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. pp. 380–386 (2016)
33. Yuan, Z., Felice, M.: Constrained grammatical error correction using statistical machine translation. In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task. pp. 52–61 (2013)
34. Zhu, J., Xia, Y., Wu, L., He, D., Qin, T., Zhou, W., Li, H., Liu, T.: Incorporating bert into neural machine translation. In: ICLR 2020 : Eighth International Conference on Learning Representations (2020)