A Research on Online Grammar Checker System Based on Neural Network Model

To cite this article: Senyue Hao and Gang Hao 2020 *J. Phys.: Conf. Ser.* **1651** 012135

A Research on Online Grammar Checker System Based on Neural Network Model

Senyue Hao[1a], **Gang Hao**[2*]

[1]School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, Shenzhen, Guangzhou, 518172, China

[2]School of Computer Science and Engineering, Tianjin University of Technology, Tianjin, Tianjin, 300384, China

[a]senyuehao@link.cuhk.edu.cn [*]Corresponding author's e-mail: gaohao2018@163.com

Abstract. Grammar error correction is one of the most important research fields in natural language processing, while grammar checker is the tool to help people correct grammatical error. We focus our work on researching a better approach to an online grammar checker system to help people correct the grammatical error in their text more precisely, user-friendly, and efficiently. This new method of online grammar checker system is based on neural network model, Transformer, and is able to detect about 25 different types of grammatical errors in the text. We did the evaluation on our new online grammar checker system, and the final experiment results proved that we have found a good approach to the online grammar checker system.

1. Introduction

People often encounter grammatical errors when they are writing essays. This problem troubles much more on non-native speakers, which usually causes people to spend a large amount of effort and time on finding and correcting the grammar error. Hence, people want to have a program that helps themselves do grammar checking automatically during writing. Grammar error correction (GEC) is one of the most important parts of natural language processing (NLP) study fields, and it is also a mature NLP task. It is mainly dealing with correcting various errors in the text, such as spelling, punctuation, grammatical, and word choice errors.

Neural network, or artificial neural network, is a popular machine learning (ML) algorithm that has sparked tremendous global interest in the last few years. Compared with conventional ML algorithms, neural networks, especially deep neural networks, have been shown to achieve significantly higher accuracies in many domains. The neural network can learn pattern recognition from big data samples through training. Then, it can do the pattern predictions for unseen samples.

Over the past decades, neural networks have re-emerged as powerful machine-learning models, yielding state-of-the-art results in NLP [15]. Several related neural network models were proposed to deal with GEC tasks, such as the encoder-decoder model [6, 9, 33], parallel iterative edit model [1], and Transformer model [7, 16, 35].

With the advent of the Internet, grammar checker is no longer limited to being an add-on program of other large edit software but can be more directly used by users through the Internet. As English is the most widely used language in the world, a good online grammar checker system will be very helpful for English learners [5]. It is already known that the neural network models have achieved brilliant results in the theoretical research of GEC tasks, scientists are curious and eager to see the results when it was applied to industrial applications.

We research a new method of online grammar checker system, which uses the state-of-art Transformer model as the backend and combines several pre-processing algorithms on the user input. Within a tolerated waiting time, given the text by people, this online system will return the corrected text that is predicted by the neural network model. In addition, we have tested our online grammar checker system on the CoNLL-2014 shared task set [24], which is one of the standard criteria for GEC task. We finally got 60.93 $F_{0.5}$ score on the test set.

2. Related Work

The mainstream of solving grammar error correction task is treated as a translation task from the language of "bad" English to the language of "good" English [9]. In the past five years, machine translation methods have been the most successful approach to GEC tasks [16]. Some work started with statistical phrase-based machine translation (SMT) methods using traditional statistical model, hidden Markov model and even utilization of neural networks [8], while sequence-to-sequence methods [34] adopted from neural machine translation (NMT) inspired more people to research the application of encoder-decoder neural network architecture in GEC.

Xie et al. [33] proposed the use of a character-level recurrent encoder-decoder network for GEC. They trained their models on the publicly available NUCLE [10] and Lang-8 corpora [23, 30], along with synthesized examples for frequent error types. They also incorporated an N-gram Language Model [24] trained on a small subset of the Common Crawl corpus (2.2 billion n-grams) during decoding to achieve an $F_{0.5}$ score of 39.97 on the CoNLL-2014 shared task test set.

After applying the attention mechanism and multiple layers of convolutions, Chollampatt and Ng [9] proposed a multilayer convolutional encoder-decoder neural network on GEC task. The encoder network is used to encode the potentially erroneous source sentence in vector space and a decoder network generates the corrected output sentence by using the source encoding. The ensemble of their model with adding a web-scale Language Model and a spelling correction component reached 54.79 $F_{0.5}$ on the CoNLL-2014 shared task test data.

Another model to solve GEC task called Parallel Iterative Edit (PIE) was raised by Abhijeet Awasthi et al. [1] in 2019. Instead of using the popular encoder-decoder model for the sequence to sequence learning and sequentially generating the tokens in the output, the PIE model generates the output in parallel, thereby substantially reducing the latency of sequential decoding on long inputs. Four advantages of this model, mentioned by Abhijeet et al. made sure that this model achieves competitive accuracy compared with the encoder-decoder model, including, 1. predicting edits instead of tokens, 2.labeling sequences instead of generating sequences, 3. iteratively refining predictions to capture dependencies, and 4. factorizing logits over edits and their token argument to harness pre-trained language models like BERT [12]. The PIE model employs several ideas to match the accuracy of sequential models despite parallel decoding: it predicts in-place edits using a carefully designed edit space, iteratively refines its predictions, and effectively reuses the BERT. By using an ensemble of the models with spell checking, pre-trained language model, and iterative refinement, the PIE model achieves 61.2 $F_{0.5}$ on the CoNLL-2014 shared task test set.

Transformer model has a strong power in handling sequence-to-sequence problems since it was first raised in 2017 [31]. It was first used in the NMT fields and scientists have adopted it into the GEC field since 2018 [18]. In the recent BEA-2019 shared task on GEC [4], the first two teams who got the highest score in the restricted task and the low-resource task are both using Transformer model as their basic algorithm. Although there is no revolution on the neural network architecture, they introduced a lot of pre-processing methods on the input text which inspires our work, since, for the online system, we are facing a very serious open vocabulary and time-saving issue.

3. Methods

We combined the Transformer model with several pre-processing methods including byte pair encoding algorithm, tokenization, and spellchecker, to build a strong and usable online grammar checker, and we believe that it is a good approach to build a good online grammar checker system, while the main aim of pre-processing is to extract the original and corrected versions of each paragraph in the input data along with the edits that transform the former into the latter [3].

3.1. Transformer Model

Transformer model was first introduced by Vaswani et al. [31] in 2017 based on the attention mechanism [2] which is aimed to solve the limitation of a simple encoder-decoder model that was first proposed in 2014 by Cho et al. [6], shown in the. The biggest limitation of the encoder-decoder model is that the only connection between encoder and decoder is a fixed-length semantic vector C. In other words, the encoder will compress the entire sequence of information into a fixed-length vector. There are two drawbacks to this. One is that the semantic vector cannot fully represent the information of the entire sequence. The other is that the information carried by the first input will be diluted by the information entered later, or covered. The longer the input sequence, the more serious this phenomenon. This makes it impossible to obtain enough information about the input sequence at the beginning of decoding, then the accuracy of decoding will naturally be discounted. The attention mechanism, which is the core of Transformer model, was proposed to solve the limitation. In short, when the encoder-decoder model generates an output, it also generates an "attention range" indicating which parts of the input sequence to focus on when outputting and then generates the next output based on the area of interest, as shown in Figure 1. Compared to PIE model, Transformer model is not only easier to implement and read but also has a good future blueprint.

Transformer model was one of the state-of-art neural machine translation architectures in 2018 [18]. As regarding GEC tasks as a machine translation task, Junczys-Dowmunt et al. [18] had adopted transfer learning and other GEC-specific adaptations on the Transformer model used in NMT, including pre-trained word-embedding, Word2vec [22], and pre-trained decoder parameters, a language model trained on the monolingual data based on condition Gate Recurrent Unit with attention mechanism. The basic architecture of Transformer model for our grammar checker system is remained the default of 6 complex self-attention blocks in the encoder and decoder, and uses the same model dimensions - embeddings vector size is 512, filter size is 2048.

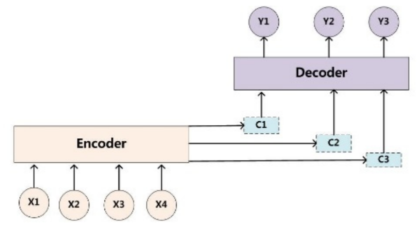

Figure 1. Encoder-decoder Model with attention mechanism

3.2. Byte Pair Encoding Algorithm

Byte Pair Encoding (BPE) is a simple form of data compression in which the most common pair of consecutive byte data is replaced with bytes that do not exist in the data [13], while de-BPE algorithm will use a replacement table to reconstruct the original data from compression. BPE allows for the representation of an open vocabulary through a fixed-size vocabulary of variable-length character sequences, making it a very suitable word segmentation strategy for neural network models. The time complexity of BPE processing on text with a length of N is O(N^2), which is suitable for our online system. In the natural language processing field, BPE is a way of pre-processing which is used to solve the open vocabulary issues or rare words problem. With the help of SentencePiece [20], which can directly train the subword models from a raw sentence, BPE will use the subword dictionary learned by corpus, stored and compressed the split unknown word in training and testing data, to enlarge the known part of the whole passage. De-BPE algorithm is used to look up the segmented text in the subword dictionary, merge the known subwords together and return the integrated sentence back to the user. BPE algorithm with SentencePiece perfectly helps to make the online grammar checker system an end-to-end system.

3.3. SpaCy Tokenization

Symbols including punctuation marks and emojis between raw sentences from user input online will definitely affect the performance of Transformer model and BPE algorithm. Therefore, a tool for pre-processing the raw text is needed. An open-sourced python library called spaCy was used to tokenize the BEA-2019 shared task text [4]. The core model used in tokenization is called *en_core_web_sm*, which is an English multi-task convolutional neural network trained on OntoNotes [32]. The genre of aimed processing text is the web written text including blogs, news, comments, and emails which are all the fields that our online grammar checker system wants to be used. The model used in the spaCy library has a 99.76 score on the tokenization accuracy [17], which is totally reliable for an online grammar checker system. The spaCy library has good documentation, nice compatibility with PyTorch and neural networks, and it is an industrial-oriented Python library.

3.4. Spellchecker

Many recent GEC systems contain a spellchecker that is not only for correcting the grammar error type "SPELL" but also for dealing with the unknown words brought by typo or misspelling. A popular open-sourced spellchecker called *hunspell* [26] was retrained by Choe et al. [7] to avoid the limitation of it, which is just primarily operating at the word level without consideration of the context. They used a gated convolutional neural network language model pre-trained on WikiText-103 dataset [21] and took the tokenized input by spaCy of the BEA-2019 shared task text to re-rank the top candidates suggested by *hunspell*. Since this pre-processing method helped Choe's team got a good score in the BEA-2019 shared task [4], we believed and adopted the language model and *hunspell* software from their work.

3.5. System Design and Implementation

The online grammar checker system is started by the command line and it will load the single trained Transformer model file before it shows the user interface to people. When users input text and click the "submit" button, the system will do the spaCy tokenization, load the spellchecker, and did the BPE algorithm and segmentation on the raw input text. Then, the system will use the loaded model to predict the correction of the input text and return the best result of the beam search back to the user. It may take a little long time if the user wants to see the different results of the system by their own customization on the beam search sizes. Figure 2 shows the flowchart of our grammar checker system procedures.

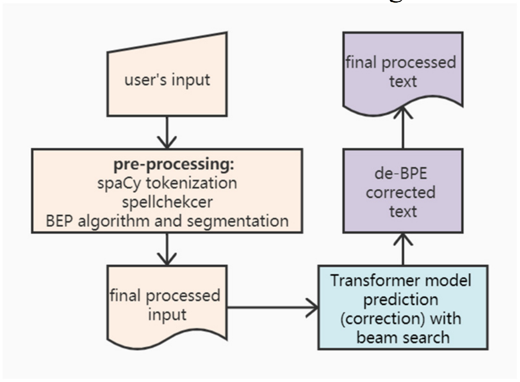

Figure 2. The flowchart of our grammar checker

As Python has its advantages on readability, inherent object-oriented architecture, and outstanding library sharing and portability, we use this programming language to train and load our model, while we continue to use the same programming language for our web interface implementation. Besides, we choose Django as our implementation framework as it has very detailed and extensive documentation, and it is very user-friendly on error handling [14].

4. Experiments and Results

4.1. Experiments Setup

4.1.1. Model Training. We first trained our backend Transformer neural network based on the BEA-2019 shared task and generated four model files which can be either separately used or as an ensemble. There are totally 25 kinds of grammar error types in this new dataset, which means that our grammar checker system should be able to detect and correct 25 kinds of grammar errors. As GEC is a sequence-to-sequence problem, we adopted fairseq [27], an open-source sequence modeling toolkit, as our training helper.

4.1.2. Prediction Parameters. After we got the four model files, we continued to use the fairseq code for loading the model file and did the prediction. However, after we did the speed test of our online grammar checker system, we found it is quite slow for using all four model files as an ensemble. Meanwhile, although we use beam search size equals to 12 when we trained the Transformer neural network, it is quite slow when we still use the same size when doing the prediction. Beam search, a heuristic searching algorithm that explores a graph by expanding the most promising node in a limited set, was used to decode the best sentence with a set beam size. When testing on the CPU, by inputting a short essay with 281 words, the total time costing with a beam size of 12 is 78 seconds, average speed 3.6 words/second. The costing time on testing the CoNLL-2014 shared task with 1,312 sentences is 6459 seconds. By changing the beam size from 12 to 1, which is the simple greedy search, the system running time speeds up. For the short essay with 281 words, the total time costing is only 28 seconds, average speed 10 words/second. The costing time on testing the CoNLL-2014 shared task is only 1677 seconds. For beam search size equals to 12, our system finally got 61.78 $F_{0.5}$ on the CoNLL-2014 shared task, and for beam search size equals to 1, the system got 60.93 $F_{0.5}$ on the same dataset. The speed test and evaluation of the system showed that change the beam search size from 12 to 1 when doing the prediction is usable and efficient for an online grammar checker system.

4.2. Evaluations and Results

To test whether this online GEC system is able to correct the grammar error in the text, we used the CoNLL-2014 shared task test, which is the most widely used dataset to benchmark GEC systems. The test set contains 1,312 English sentences with totally 28 kinds of grammatical errors. By using the CoNLL-2014 shared task, we may test the robustness of this online GEC system. MaxMatch (M^2) Scorer is often used to generate the precision, recall, and $F_{0.5}$ measurements [11] for this test set. The whole experiment result is shown in Table 1.

Table 1. Comparison of various GEC models

	Precision	**Recall**	**$F_{0.5}$**
Chollampatt &Ng (NUS Grammar Checker)	65.49	33.14	54.79
Zhao et al.	71.57	38.65	61.15
Awasthi et al.	66.14	43.04	59.74
Our System (Beam Size = 1)	68.43	42.36	60.93
Our System (Beam Size = 12)	69.62	42.58	61.78

For M^2 Scorer, given a set of n sentences, where g_i is the set of gold-standard edits for sentence i, and e_i is the set of system edits for sentence i, recall, precision, and $F_{0.5}$ are defined as follows:

$$\text{Recall} = \frac{\sum_{i=1}^{n}|g_i \cap e_i|}{\sum_{i=1}^{n}|g_i|} \quad (1)$$

$$\text{Precision} = \frac{\sum_{i=1}^{n}|g_i \cap e_i|}{\sum_{i=1}^{n}|e_i|} \quad (2)$$

$$F_{0.5} = \frac{(1+0.5)^2 \times Recall \times Precision}{Recall + 0.5^2 \times Precision} \quad (3)$$

where the intersection between g_i and e_i for sentence i is defined as:

$$g_i \cap e_i = \{e \in e_i \mid \exists g \in g_i, match(g, e)\} \quad (4)$$

5. Discussion and Conclusion

From the experiment, we can know that though the recall of this online system is a little bit high, the precision and $F_{0.5}$ are good enough. In addition, we can see that the system with beam search size of 12 gets better results on precision and $F_{0.5}$ than the system with a beam search size of only 1, while the system with beam search size of 1 is much faster than the system with beam search size of 12. Exchanging the speed with the accuracy in this online circumstance seems reasonable. Therefore, the online grammar checker system chooses beam search size of 1 as its final version. We successfully implement an online neural-network-based grammar checker system. Based on the Transformer model, spaCy tokenization, spellchecker, and BPE segmentation algorithm, it got 60.93 $F_{0.5}$ on the CoNLL-2014 shared task with a beam search size of 1. Meanwhile, by testing the web interface with a short paragraph, the speed of prediction is decent, with a short essay, 10 words/second, and for a long essay, it will perform better. This system is still a pre-release version. It started by the command line and not run in conjunction with Apache. Besides, this online neural-network-based grammar checker system should do the grammar checking more real-time like Grammarly [25], one of the most popular grammar checker systems in the world, and display different grammar errors with different colors.

References

[1] Awasthi, A., Sarawagi, S., Goyal, R., Ghosh, S., & Piratla, V. (2019). Parallel iterative edit models for local sequence transduction. arXiv preprint arXiv:1910.02893.

[2] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[3] Bryant, C., & Felice, M. (2016). Issues in preprocessing current datasets for grammatical error correction (No. UCAM-CL-TR-894). University of Cambridge, Computer Laboratory.

[4] Bryant, C., Felice, M., Andersen, Ø. E., & Briscoe, T. (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 (pp. 52-75).

[5] Bustamante, F. R., & Leo´n, F. S. (1996). GramCheck: A grammar and style checker. arXiv preprint cmp-lg/9607001.

[6] Cho, K., Van Merrie¨nboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[7] Choe, Y. J., Ham, J., Park, K., & Yoon, Y. (2019). A neural grammatical error correction system built on better pre-training and sequential transfer learning. arXiv preprint arXiv:1907.01256.

[8] Chollampatt, S., Taghipour, K., & Ng, H. T. (2016). Neural network translation models for grammatical error correction. arXiv preprint arXiv:1606.00189.

[9] Chollampatt, S., & Ng, H. T. (2018). A multilayer convolutional encoder-decoder neural network for grammatical error correction. In Thirty-Second AAAI Conference on Artificial Intelligence.

[10] Dahlmeier, D., Ng, H. T., & Wu, S. M. (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 (pp. 22-31).

[11] Dahlmeier, D., & Ng, H. T. (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 (pp. 568-572).

[12] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre- training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[13] Gage, P. (1994). A new algorithm for data compression. C Users Journal, 12(2), 23-38.
[14] Ghimire, D. (2020). Comparative study on Python web frameworks: Flask and Django.
[15] Goldberg, Y. (2016). A primer on neural network models for natural language processing. Journal of Artificial Intelligence Research, 57, 345- 420.
[16] Grundkiewicz, R., Junczys-Dowmunt, M., & Heafield, K. (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 (pp. 252-263).
[17] Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language under- standing with Bloom embeddings, convolutional neural networks and incremental parsing.
[18] Junczys-Dowmunt, M., Grundkiewicz, R., Guha, S., & Heafield, K. (2018). Approaching neural grammatical error correction as a low- resource machine translation task. arXiv preprint arXiv:1804.05940.
[19] Karyuatry, L. (2018). Grammarly as a Tool to Improve Students' Writing Quality: Free Online-Proofreader across the Boundaries. JSSH (Jurnal Sains Sosial dan Humaniora), 2(1), 83-89.
[20] Kudo, T., & Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226.
[21] Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2016). Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843.
[22] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
[23] Mizumoto, T., Hayashibe, Y., Komachi, M., Nagata, M., & Matsumoto, Y. (2012). The effect of learner corpus size in grammatical error correction of ESL writings. In Proceedings of COLING 2012: Posters (pp. 863-872).
[24] Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., & Bryant, C. (2014). The CoNLL-2014 shared task on grammatical error correction. In Proceedings of the Eighteenth Conference on Computa- tional Natural Language Learning: Shared Task (pp. 1-14).
[25] Omelianchuk, K., Atrasevych, V., Chernodub, A., & Skurzhanskyi, O. (2020). GECToR–Grammatical Error Correction: Tag, Not Rewrite. arXiv preprint arXiv:2005.12592.
[26] Ooms, J. (2018). hunspell: High-performance stemmer, tokenizer, and spell checker. Pobrane z https://CRAN. R-project. org/package= hun- spell.
[27] Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., ... & Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. arXiv preprint arXiv:1904.01038.
[28] See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368.
[29] Stutz, M. (2004). The Linux Cookbook: Tips and Techniques for Everyday Use. No Starch Press.
[30] Tajiri, T., Komachi, M., & Matsumoto, Y. (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) (pp. 198-202).
[31] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
[32] Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., ... & El-Bachouti, M. (2013). Ontonotes release 5.0 ldc2013t19. Linguistic Data Consortium, Philadelphia, PA, 23.
[33] Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., & Ng, A. Y. (2016). Neural language correction with character-based attention. arXiv preprint arXiv:1603.09727.
[34] Yuan, Z., & Briscoe, T. (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 (pp. 380-386).
[35] Zhao, W., Wang, L., Shen, K., Jia, R., & Liu, J. (2019). Improving gram- matical error correction

via pre-training a copy-augmented architecture with unlabeled data. arXiv preprint arXiv:1903.00138.