



## Article

# Applying a Hybrid Sequential Model to Chinese Sentence Correction

Jun Wei Chen <sup>1</sup>, Xanno K. Sigalingging <sup>1,\*</sup>, Jenq-Shiou Leu <sup>1</sup>  and Jun-Ichi Takada <sup>2</sup> 

<sup>1</sup> Department of Electronic and Computer Engineering, College of Electrical Engineering and Computer Science, National Taiwan University of Science and Technology, Taipei City 10607, Taiwan; m10602138@mail.ntust.edu.tw (J.W.C.); jsleu@mail.ntust.edu.tw (J.-S.L.)

<sup>2</sup> Department of International Development Engineering, Graduate School of Science and Engineering, Tokyo Institute of Technology, 2-12-1-S6-4, O-okayama, Meguro-ku, Tokyo 152-8550, Japan; takada@ide.titech.ac.jp

\* Correspondence: d10502810@mail.ntust.edu.tw

Received: 12 November 2020; Accepted: 23 November 2020; Published: 25 November 2020



**Abstract:** In recent years, Chinese has become one of the most popular languages globally. The demand for automatic Chinese sentence correction has gradually increased. This research can be adopted to Chinese language learning to reduce the cost of learning and feedback time, and help writers check for wrong words. The traditional way to do Chinese sentence correction is to check if the word exists in the predefined dictionary. However, this kind of method cannot deal with semantic error. As deep learning becomes popular, an artificial neural network can be applied to understand the sentence's context to correct the semantic error. However, there are still many issues that need to be discussed. For example, the accuracy and the computation time required to correct a sentence are still lacking, so maybe it is still not the time to adopt the deep learning based Chinese sentence correction system to large-scale commercial applications. Our goal is to obtain a model with better accuracy and computation time. Combining recurrent neural network and Bidirectional Encoder Representations from Transformers (BERT), a recently popular model, known for its high performance and slow inference speed, we introduce a hybrid model which can be applied to Chinese sentence correction, improving the accuracy and also the inference speed. Among the results, BERT-GRU has obtained the highest BLEU Score in all experiments. The inference speed of the transformer-based original model can be improved by 1131% in beam search decoding in the 128-word experiment, and greedy decoding can also be improved by 452%. The longer the sequence, the larger the improvement.

**Keywords:** BERT; transformer; RNN; deep learning

## 1. Introduction

As communication and transportation technology advances, communicating with foreigners is more common than it used to be. Chinese has become one of the most popular languages. In a globalization era, speaking a few foreign languages is a common thing. However, it is sometimes hard to tell if our grammar is correct; it would be wonderful if there would be a system that can automatically correct sentences.

The traditional way may correct sentences is with a predefined dictionary. Although this can be easily scaled up because of its low computational cost, it is challenging to correct semantic errors and grammatical errors without capturing sentence-level information.

With the deep learning method, the learnable neural network is able to capture context and sentence-level information to correct semantic errors and grammatical errors. Several researches have developed methods to use machine learning for Chinese language correction [1–3]. However, deep learning methods have intrinsic disadvantages. With higher computational costs, it is hard to scale up to commercial services.

We introduce hybrid models as the solution, using Bidirectional Encoder Representations from Transformers (BERT) [4] and Transformer [5] as the baseline, the hybrid model speeds up the inference speed, without reducing the correctness.

## 2. Existing Work

Several researches have been done which in turn are able to solve problem of capture context and sentence-level information to correct semantic and grammatical errors. Some of the researches are:

### 2.1. Sequence to Sequence (Seq2Seq)

Our task to correct an incorrect Chinese sentence can be viewed as a sequence transformation problem, transforming an error sequence to a correct sequence. Sequence to sequence [6] is a model which can be used for sequence transformation [7–9]. While encoder captures the concept of the input sequence, decoder generates sentences corresponding to the concept that encoder have captured.

### 2.2. Recurrent Neural Network

Using a recurrent neural network (Figure 1) to process sequential data has become a tradition in the deep learning field. Many models are based on recurrent neural networks. Its chain-like structure and hidden state mechanism make it possible to process sequential data [10]. However, the chain-like structure also leads to harder parallelization and may lose information after many steps of hidden state updates.

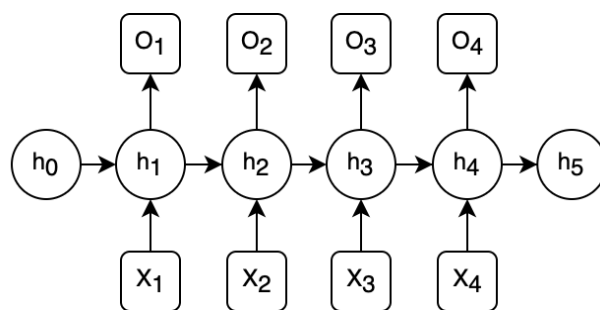


Figure 1. Recurrent Neural Network.

### 2.3. Long Short-Term Memory (LSTM)

Vanilla recurrent neural network cells update the hidden state at each step thoroughly [11], even if there is a piece of information for which it is essential to remain in the hidden state. LSTM [12] uses multiple learnable gates to control the update flow, to decide which information should be cleaned, remained, and written into the hidden state. This mechanism makes the hidden state update more efficient and loses less information. This can be seen in Figure 2,  $\sigma$  means sigmoid.

### 2.4. Gated Recurrent Unit (GRU)

Although LSTM can deal with most problems, the 3 learnable gates mean a more complicate structure, which may suffer from lower training speed, higher memory cost, and being harder to train. GRU [13],

with only two gates, update gate and reset gate (Figure 3), which is more lightweight than LSTM. However, some research [14] shows that GRU may have comparable results to LSTM in some tasks.

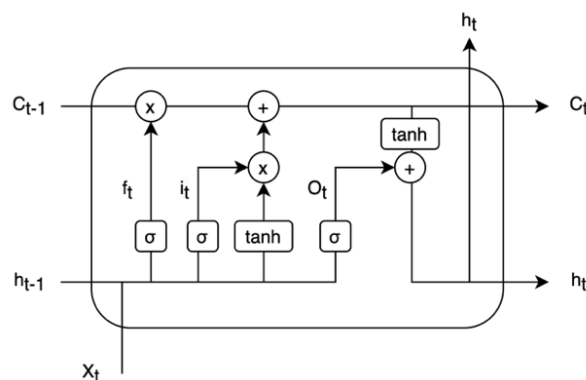


Figure 2. Long Short-Term Memory.

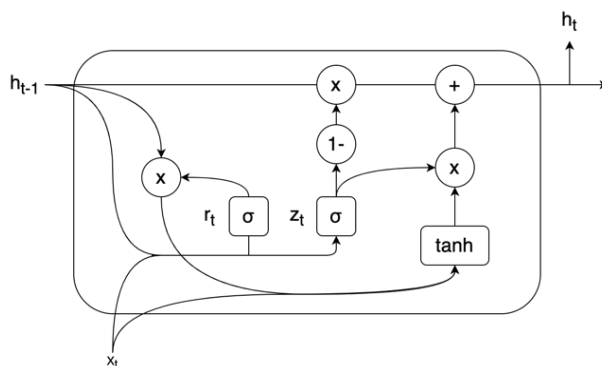


Figure 3. Gated Recurrent Unit.

## 2.5. Transformer

Transformer as a novelty architecture, which can also be used in the sequential task, but its core concept is far from that of the recurrent neural network. There is no hidden state and chain-like structure in Transformer, but only self-attention. Self-attention is a method that allows each token in the sequence to calculate the correlation with all other tokens and extract relevant and useful information at the same time. The graph for self-attention can be seen in Figure 4. It is based on the attention mechanism [15]. The most significant advantage of self-attention is that the distance between each token in the sequence is fixed, and it will not be affected by the length of the input sequence. It can also prevent a network from losing some information while processing long sequences such as RNN, thereby reducing training time and gradient vanishing.

The disadvantage of self-attention is that the computational cost will be relatively large. Without having the chain-like structure as RNN, a Transformer composed of self-attention does not need to wait for the completion of the previous state to process the next state, like RNN does, which makes the Transformer parallelizable.

## 2.6. BERT

BERT (Bidirectional Encoder Representations from Transformers), as its name suggests, is a pre-trained language model based on a Transformer encoder, which can be adopted to many different natural language processing tasks.

The reason to use a pre-trained language model is that sometimes we may lack data in some specific task. Because the model is pre-trained on large-scale datasets first, it may compensate the for lack of data, and it can also speed up the training since the pre-training-made model has already learned the language modeling.

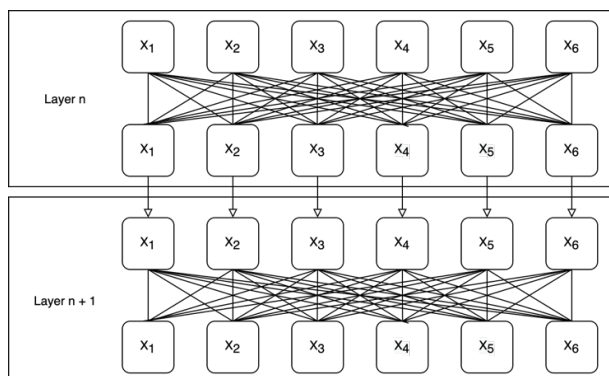


Figure 4. Self-attention.

### 3. Proposed Method

The correction system architecture that we implemented is shown in Figure 5. First, the text to be corrected will be used as input into the system, and it must be pre-processed before performing other operations. Because the correction system cannot understand the human-readable text, the text must be converted into a format that can be read by the machine before it can be input to Seq2Seq for correction. When the encoder understands the sentence's meaning, it will correct the errors and output it through different decoding methods according to the training phase and the inference phase.

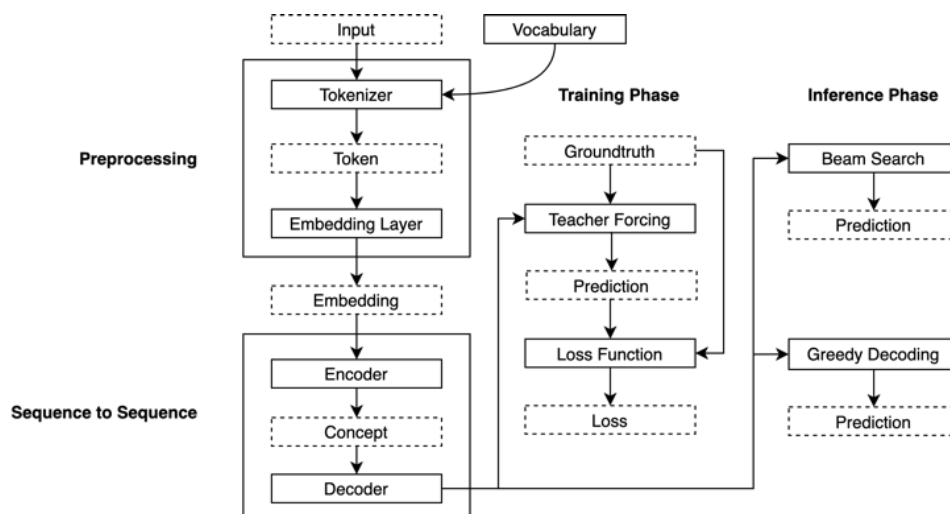


Figure 5. System Architecture.

#### 3.1. Preprocessing

The preprocessing can be divided into three parts: tokenizer, vocabulary, and embedding layer. The tokenizer, dictionary, and embedding layer are inseparable, and the dictionary is usually used to determine how the tokenizer should segment the sentence.

### 3.2. Vocabulary

Vocabulary is essential in a natural language processing system, and the use of vocabulary can be divided into two categories, word-based vocabulary, and character-based vocabulary. We use character-based vocabulary, because character-based vocabulary is more flexible, unlike word-based vocabulary, which is not prone to encounter out-of-vocabulary words. Because we were implementing a sentence correction system, the input is more likely to contain unknown words than other systems.

### 3.3. Tokenizer

The purpose of the tokenizer is to split a sentence into multiple words or combinations of characters. However, it must be used with a vocabulary in practical applications. The type of tokenizer is determined according to the characteristics of the dictionary. Because we use a character-based vocabulary, the tokenizer also uses a character-based tokenizer.

### 3.4. Embedding Layer

The embedding layer converts the characters or words that have been segmented by the tokenizer into a machine-readable format. The converted product is called embedding or word embedding, which represents words and characters with a high dimensional vector, and can also express the relationship between each words and characters.

### 3.5. Language Model

The language model is a way to represent the probability of a sentence through a distribution. Without a good language model, a good natural language processing system cannot be achieved. We use a fine-tuning-based language model. Fine-tuning-based refers to the second training on a pre-trained language model so that the language model can be better applied to different natural language process tasks, and the fine-tuning-based model we used in the experiment is BERT. BERT is a language model and also an encoder. Therefore, for the rest of the non-BERT-based encoders in the experiments, we still use the BERT vocabulary, but the language model itself has not been pre-trained.

### 3.6. Encoder Properties

Since the encoder's input sequence is known in both the training phase and the inference phase, the encoder can be parallelized in the training phase. The inference phase is roughly the same, and mainly depends on the model's architecture. As is the case with Transformer architecture, it is parallelizable because of the self-attention mechanism. In contrast, the encoder-based on recurrent neural networks are more difficult to parallelize. Because of the existence of the hidden state mechanism, the encoder must wait to complete the previous hidden state before proceeding to the next one. The model architecture mainly determines the directionality of the encoder. For example, the structure of the recurrent neural network is uni-directional by default. However, the bi-directional encoding can be achieved by using two encoders from different directions, while the Transformer encoder is bi-directional by default. The Encoder properties are shown in Tables 1 and 2.

**Table 1.** Encoder Properties at Different Phase.

Phase	Parallelizable	Direction
Training	Depends on model	Depends on model
Inference	Depends on model	Depends on model

### 3.7. Decoder Properties

The parallelization of the decoder may be different during the training phase and the inference phase. The decoder cannot be parallelized at the inference phase as it has to wait for the previous output to be complete. However, the decoder acts differently at the training phase depending on its architecture and training method; it still has to wait for previous output at the training phase. However, it is possible to undertake parallel training with the specific training method.

Teacher forcing is a method that given the decoder groundtruth sequence at the training phase, so the decoder only needs to predict one step. Because of the existence of groundtruth, the decoder can see through the complete sequence, which makes parallel training possible. However, RNN needs to wait for the hidden state at each step, so it is still challenging to achieve parallel training even if teacher forcing is used. As for the directional decoding, the decoder cannot know the complete sequence in advance, so the Transformer-based decoder can only perform uni-directional decoding. Although teacher forcing is likely to let the decoder see the complete sequence, it will lead to inconsistency between the training phase and the inference phase. The Decoder properties are shown in Tables 3 and 4.

### 3.8. Analysis

BERT is a pre-trained language model based on Transformer, so it inherits both the advantages and disadvantages of Transformer. We found that a Transformer-based sequence to sequence model has high performance in Chinese sentence correction, but it is very slow at the inference phase. By inspecting the property of the encoder and decoder, we found that the Transformer-based sequence to sequence model is slow at the decoder.

The Transformer-based decoder is slow at the inference phase because it is based on self-attention. With self-attention, the Transformer does not have the chain-like structure of the RNN-based model, which makes the Transformer powerful and parallelizable but also suffers from higher computational complexity.

At the inference phase, the encoder knows the complete input sequence in advance, which makes Transformer encoder parallelizable, but the decoder generates the next token based on the previous token, which makes the decoding process hard to parallelize.

Though the Transformer encoder has higher computational complexity, we found that parallelization has a significant effect on reducing execution time. However, the Transformer decoder has no such feature, which slows down the whole sequence transformation process.

To speed up the inference speed of Chinese sentence correction, we introduce the hybrid model by combining the BERT and RNN-based models, which speed up the inference speed but still preserve the Transformer-based model's performance.

**Table 2.** Model Encoder Properties.

	<b>RNN</b>	<b>Transformer</b>
Direction	Uni-Direction	Bi-Direction
Parallelizable	False	True
Performance	Low	High
Computational Cost	Low	High

**Table 3.** Decoder Properties at Different Phase.

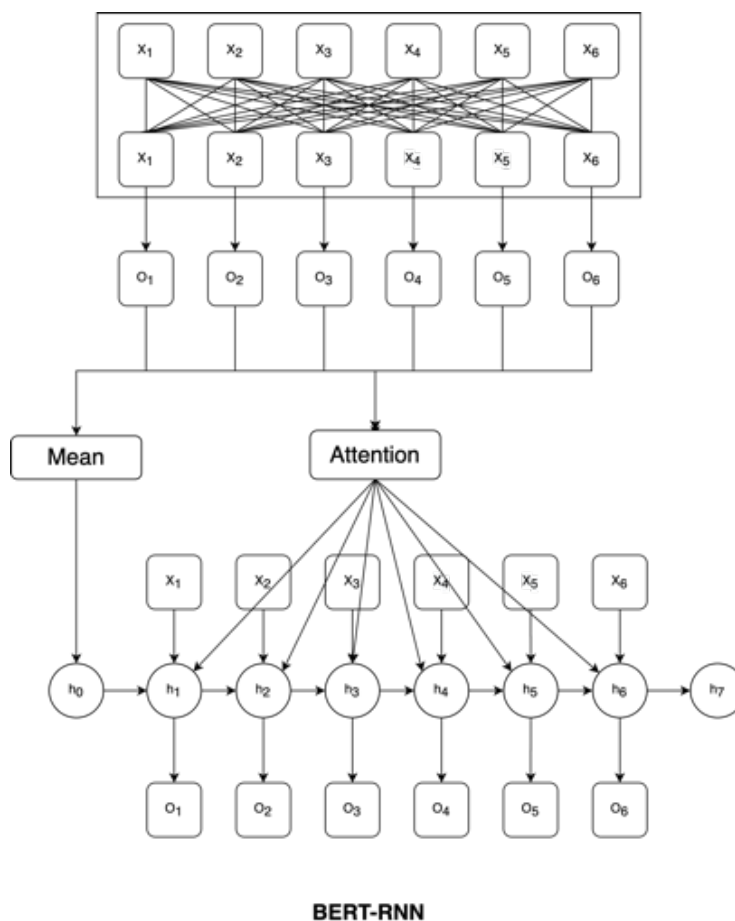
Phase	Parallelizable	Direction
Training	Depends on training method, model	Uni-Direction
Inference	False	Uni-Direction

**Table 4.** Model Decoder Properties.

	RNN	Transformer
Direction	Uni-Direction	Uni-Direction
Parallelizable	False	False
Performance	Low	High
Computational Cost	Low	High

### 3.9. Hybrid Architecture, BERT-RNN

The Transformer-based model is very different from the RNN-based model, so they cannot be combined directly. We usually have to initialize the decoder hidden state for RNN-based sequence to sequence model. There are many ways to initialize the decoder's hidden state, such as using another neural network to predict the initial state. However, we use the most straightforward method, averaging BERT's output to avoid the extra computation, as seen in Figure 6.

**Figure 6.** Bidirectional Encoder Representations from Transformers (BERT)-Recurrent Neural Network (RNN).

### 3.10. BERT-LSTM

As shown in Figure 7, the first hybrid model is BERT-GRU. It concatenates the BERT encoder and GRU decoder, containing the fast and high-performance characteristics of the BERT encoder, and the lightweight, adequate performance of the GRU decoder, which is specialized in improving inference speed.

### 3.11. BERT-GRU

The second hybrid model is BERT-LSTM (Figure 8). It concatenates the BERT encoder and LSTM decoder. LSTM has more parameters than the GRU model, but theoretically, it has better performance. Therefore, this hybrid model is specialized in improving performance rather than improving inference speed compared to BERT-GRU.

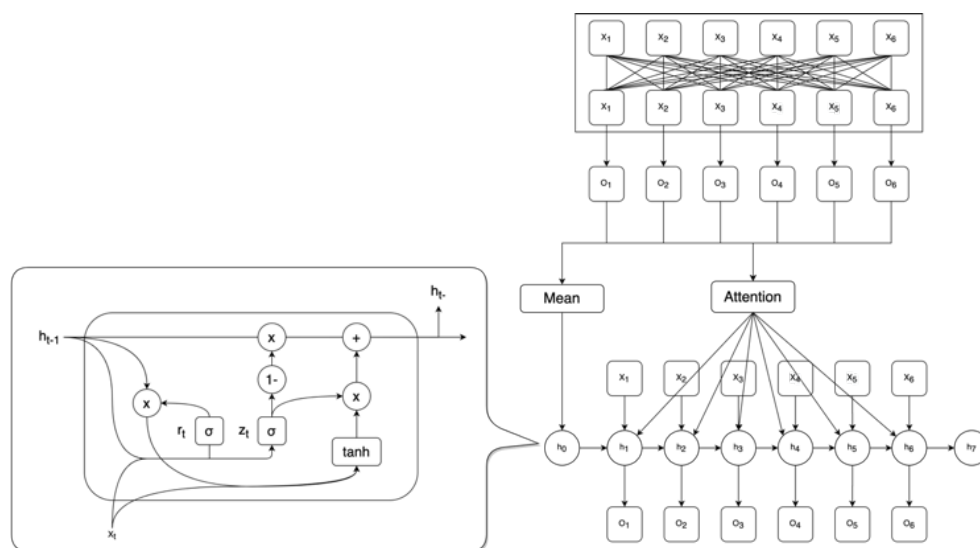


Figure 7. BERT-Gated Recurrent Unit (GRU).

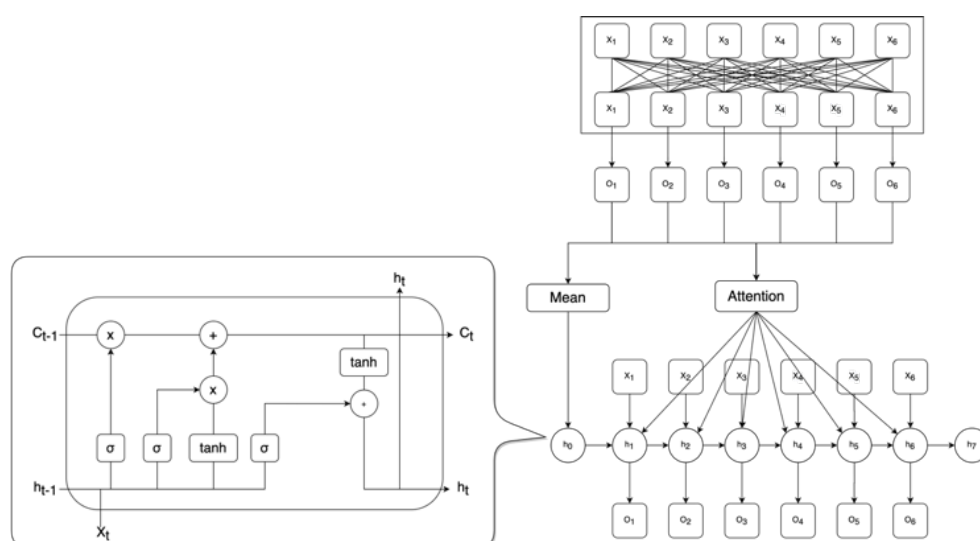


Figure 8. BERT-Long Short-Term Memory (LSTM).



### 3.12. Training Methods

The simplest way of training sequential tasks is by generating the target sequence directly and calculating the loss function according to the inference phase. This method is called free running (Figure 9). Its most significant disadvantage is that training is hard to converge because sequential tasks are a structured prediction problem. That is, the outputs are dependent on each other. In the beginning, the correctness of the prediction from the sequential model is still low, so if the previous step prediction is wrong, it may also turn the next step prediction into a wrong prediction. The accumulation of errors will lead to larger oscillations during training, which will take a long time to converge. Besides, the gradient will have to be calculated back along the time axis, which requires a larger amount of memory.

Therefore, in the training phase, we will use teacher forcing for training (Figure 10). Teacher forcing is a training technique proposed to solve the training instability. We will shift groundtruth one step to the right and add a special symbol to indicate the start signal. BOS in Figure 10 stands for the beginning of the sequence, so for the Decoder, the input at each step would be correct. Thus, the error mentioned above will not occur in teacher forcing. However, this training method also has side effects, that is, the sequential model trained with teacher forcing just has to learn to predict one step, and does not consider the output of the next few steps and the overall structure of the sentence. Such a problem is called exposure bias. The solution we use is beam search, which will be introduced in detail in the following sections.

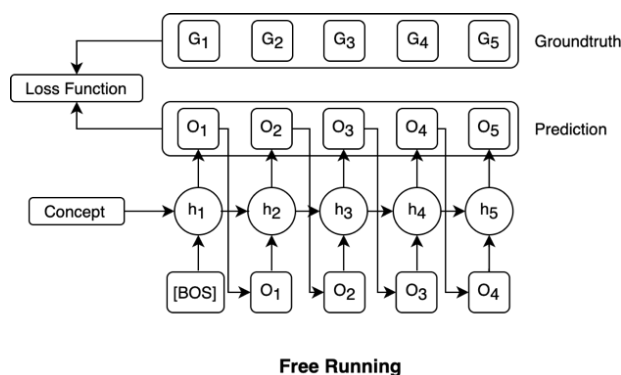


Figure 9. Free Running.

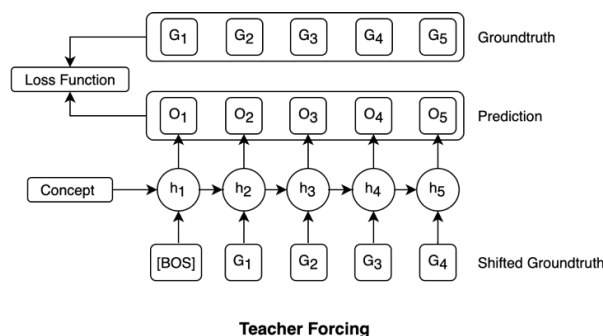


Figure 10. Teacher Forcing.

### 3.13. Greedy Decoding

Greedy decoding is the most basic decoding method in sequential tasks. At each step of decoding, the token with the highest probability is selected as the output. It is called greedy decoding because the algorithm always selects the best option while decoding. The decoder does not consider the long-term

future with more foresight. However, in the long run, the best choice now at each step may not necessarily be the best sequence. This problem is called exposure bias, and one of the reasons for exposure bias is teacher forcing.

Although teacher forcing makes the training process relatively stable, it causes the sequential model learning only to predict one step during training and will not consider the long-term future, which leads to inconsistency between the inference phase and training phase. One of the solutions is beam search.

### 3.14. Beam Search

Beam search, as shown in Figure 11, is a method of searching existing sentences without training and trying to find a better sentence than greedy decoding. When beam search is decoding, it selects multiple outputs with the highest probability as candidate outputs for each step, forming a tree with multiple branches. The number of candidate outputs selected at each step is called beam size. The decoder generates multiple branches at each step until each branch receives the end signal, or the sentence exceeds the length limit and then selects a branch with the highest probability as the final output. Although beam search can choose a better answer, it requires a more extensive computation than greedy decoding. Greedy decoding can be regarded as a beam search with a beam size of 1, which means that the best output is selected as a candidate each time.

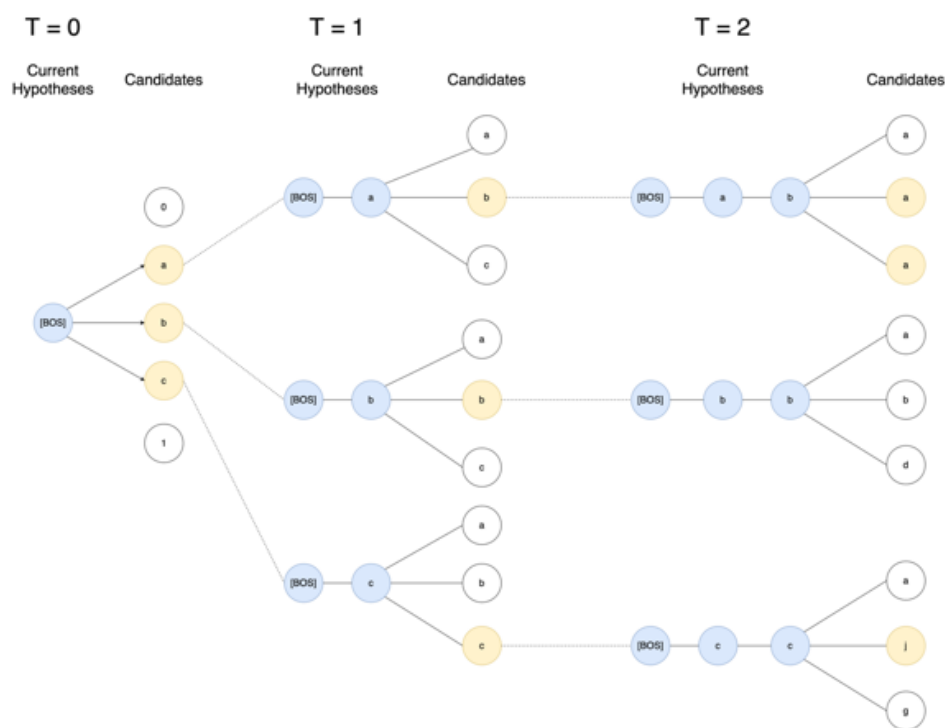


Figure 11. Beam Search.

## 4. Experiment

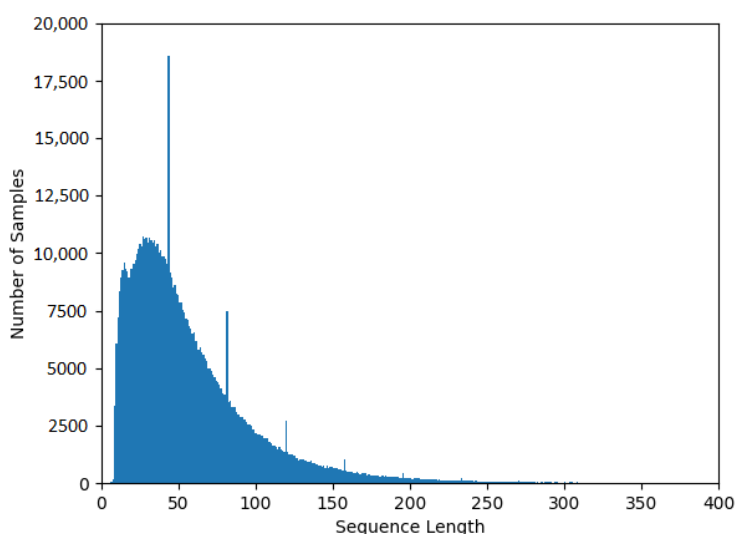
For our experiment, we prepare the dataset, environment, settings, and certain metrics. The subsection below explains the experiment thoroughly.

#### 4.1. Dataset

We use NLPCC 2018 grammatical error correction (GEC) dataset [16] to do the experiment. However, groundtruth of GEC's testing set is not available, so we split the GEC's training set into a custom training set and custom testing set. We perform the experiment on three different sequence lengths, 25, 50 and 128 respectively as Table 5 shows. We chose the lengths based on presumption the sequences lengths are able to reflect the relation between length and the results, according to the sequence length distribution seen in Figure 12.

**Table 5.** Dataset Distribution.

Length	Train Set	Test Set
25	686,130	49,157
50	1,056,324	69,446
128	1,093,564	71,653



**Figure 12.** Sequence Length Distribution.

#### 4.2. Environment

The experiment was done on a single machine using Arch Linux as the operating system, Intel i7 6700 as CPU, Nvidia 1080 Ti as GPU. We use MXNet [17], GluonNLP [18] as framework.

#### 4.3. Experiment Settings

We use Adam (Adaptive Moment Estimation) [19] as the neural network optimizer in the experiment. Adam will refer to the previous update direction when updating the parameters, and adjust the learning rate according to the gradient. The learning rate in the experiment is set to 0.0001. We use both greedy decoding (GD) and beam search (BS) in the experiment.

#### 4.4. Vocabulary Setup

As we use BERT in the experiment, the source vocabulary we followed the default BERT vocabulary, including simplified Chinese character, traditional Chinese character, lowercase English character, and some punctuation marks, 21,128 characters in total.

In the experiment, we convert all simplified Chinese characters into traditional Chinese characters, which means our target vocabulary only has to contain traditional Chinese characters, English characters, punctuation marks, 10,991 characters in total, nearly half of the source vocabulary, which also reduces the memory usage and training time.

#### 4.5. Evaluation Metric

We use BLEU (Bilingual Evaluation Understudy Score) [20] to evaluate the performance. It was originally used as a tool to evaluate machine translation. The principle is to compute the overlap ratio of consecutive characters between prediction and groundtruth. In the experiment, we use the NLTK [16] package, and  $N = 4$ , which calculates a single character's overlap rate to the overlap rate of four consecutive characters.

#### 4.6. Model Performance Comparison

The naming convention of the model is encoder-decoder. For example, GRU-GRU means that the model's encoder is GRU, and the decoder is also GRU. The unit of inference speed and training speed is sample per second.

From the results above, we can see that the pure RNN-based model has lower performance than the pure Transformer-based model, but it has faster inference speed. However, our hybrid model has similar performance with the Transformer-based model, but faster inference than the pure RNN-based model and our hybrid model BERT-GRU showed the best performance and fastest inference speed in 3 experiments, no matter whether it used beam search or greedy decoding.

From the pure RNN-based model's perspective, replacing the RNN-based encoder with a Transformer-based encoder accelerates the encoding process for the inference phase and training phase, which also improves performance. However, from the perspective of the pure Transformer-based model, replacing the Transformer-based decoder with an RNN-based decoder accelerates the inference phase decoding process. However, it slows down decoding process at the training phase.

We use teacher forcing at the training phase, which is a training method for a given groundtruth to the decoder to know the entire decoding target in advance, which makes decoder possible to parallelize, without teacher forcing. The decoder has to wait for the previous token to be predicted, but the RNN-based model also has to wait for the previous hidden state, so it still cannot be parallelized, even if training with teaching forcing. As a result, it slows down the training speed of the hybrid model.

The experimental result also shows that our hybrid model speeds up more while we turn our pure Transformer-based model into the hybrid model using a beam search. The reason for this is that the beam search has more load at the decoder, so when we replace the Transformer-based decoder with a lightweight RNN-based decoder, it speeds up more.

#### 4.7. Experiment Comparison

The data used in the three sets of experiments are not precisely the same, so the following comparison will focus on the improvement of speed.

Figure 13 shows the percentage increase in the hybrid model's inference speed for the RNN-Based original model under different lengths. The vertical axis represents the percentage increase in the inference speed, and the horizontal axis represents experiments of different lengths. We can see that both the beam search and the greedy decoding speed has been improved. The longer the sequence, the more the improvement. Since the RNN-based encoder is replaced with the Transformer-based encoder, the beam search which focuses on the decoding stage has poorer improvement. In contrast, using greedy decoding, the longer the sequence, the higher the ratio of improvement. Therefore, Figure 13 shows that the two lines using greedy decoding are non-linear.

Figure 14 shows the percentage increase in the inference speed of the hybrid model for the Transformer-based original model under different sequence lengths. The vertical axis represents the percentage increase in the inference speed, and the horizontal axis represents experiments of different lengths. Both the beam search and greedy decoding's inference speed are improved. The longer the sequence, the more the improvement, as we replace the Transformer-Based decoder with the RNN-based decoder, so the beam search, which focuses on the decoding stage, has a greater improvement. Moreover, Figure 14 does not have apparent non-linear improvement, as shown in Figure 13. The main reason for this is that the decoder itself cannot be parallelized during inference. Therefore, the increase of speed relies on GRU and LSTM's lightweight, while in Figure 13 we replace the non-parallelizable RNN-based encoder with the parallelizable BERT encoder, and the non-linear improvement is more significant. Detailed results can be seen in Tables 6–8.

Table 6. Result of Experiment 25.

Model	BLEU Score-BS	Inference Speed-BS	BLEU Score-GD	Inference Speed-GD	Training Speed
GRU-GRU	0.7591	447.246	0.7553	793.454	221.48
LSTM-LSTM	0.7549	425.297	0.7490	692.05	197.106
TRANS-TRANS	0.7645	115.955	0.7596	410.753	362.085
BERT-TRANS	0.7667	112.603	0.7625	403.235	359.804
BERT-GRU	0.7676	535.172	0.7627	1038.811	277.962
BERT-LSTM	0.7669	504.521	0.7617	952.377	261.673

Table 7. Result of Experiment 50.

Model	BLEU Score-BS	Inference Speed-BS	BLEU Score-GD	Inference Speed-GD	Training Speed
GRU-GRU	0.7975	214.261	0.7953	417.462	132.446
LSTM-LSTM	0.7970	200.268	0.7945	363.304	114.758
TRANS-TRANS	0.7995	30.857	0.7966	149.292	264.928
BERT-TRANS	0.7997	33.657	0.7959	141.639	265.218
BERT-GRU	0.8021	272.526	0.7986	579.366	176.705
BERT-LSTM	0.8017	243.564	0.7982	533.153	162.79

Table 8. Result of Experiment 128.

Model	BLEU Score-BS	Inference Speed-BS	BLEU Score-GD	Inference Speed-GD	Training Speed
GRU-GRU	0.8042	111.094	0.8022	233.648	55.635
LSTM-LSTM	0.8034	100.455	0.8010	203.333	47.137
TRANS-TRANS	0.8068	10.91	0.8041	68.936	142.064
BERT-TRANS	0.8069	12.362	0.8039	70.803	142.21
BERT-GRU	0.8092	152.276	0.8059	391.5	76.775
BERT-LSTM	0.809	130.146	0.8057	359.848	69.633

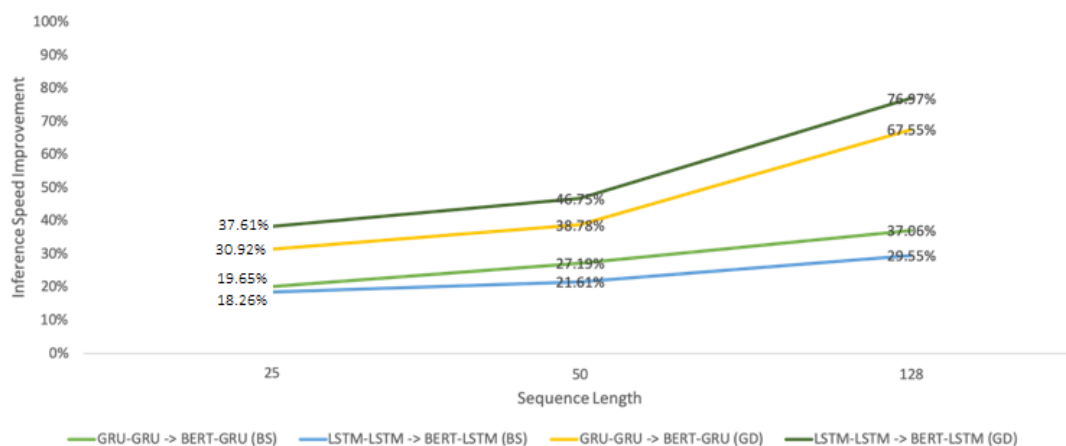


Figure 13. Hybrid model improvement with respect to RNN-based model.

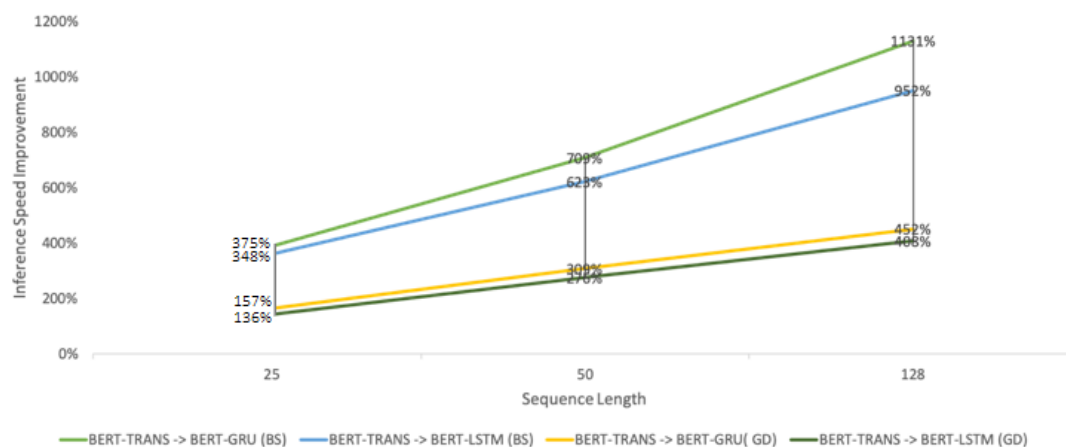


Figure 14. Hybrid model improvement with respect to Transformer-based model.

## 5. Conclusions

As Transformer architecture is gradually being popular, the field of natural language processing has seen the deprecation of RNN series models. By inspecting the advantages and disadvantages of RNN and Transformer, we introduce a hybrid model with faster inference speed and better performance.

In addition to the pre-training of the BERT encoder itself, the Transformer encoder also has the characteristics of parallelization, which makes up for the shortcomings of GRU and LSTM as encoders in terms of speed and performance. In contrast, GRU and LSTM as decoders make up for the slower inference speed of the Transformer-based model, so we can say that they compensate for each other's shortcomings. Through multiple sets of experiments, we have also proved that this combination can indeed be applied to Chinese sentence correction. Among them, BERT-GRU has obtained the highest BLEU Score in all experiments. The inference speed of the Transformer-based original model can be improved by 1131% in beam search decoding in the 128-word experiment, and greedy decoding can also be improved by 452%. The longer the sequence, the larger the improvement.

**Author Contributions:** Conceptualization, J.W.C. and X.K.S.; Methodology, J.W.C.; Project administration, J.-S.L. and J.-I.T.; Software, J.W.C. and X.K.S.; Supervision, J.-S.L. and J.-I.T.; Visualization, J.W.C.; Writing—original draft, X.K.S.; Writing—review and editing, J.-S.L. and J.-I.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study was supported by a collaborative research project between the National Taiwan University of Science and Technology (Taiwan-Tech) and the Tokyo Institute of Technology (Tokyo-Tech), funded under number Grant TIT-NTUST-107-05.

**Acknowledgments:** The authors gratefully acknowledge the support extended by the Taiwan Tech-Tokyo Tech Joint Research Program, under number Grant TIT-NTUST-107-05.

**Conflicts of Interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

1. Huang, C.M.; Wu, M.C.; Chang, C.C. Error Detection and Correction Based on Chinese Phonemic Alphabet in Chinese Text. In Proceedings of the International Conference on Modeling Decisions for Artificial Intelligence, Kitakyushu, Japan, 16–18 August 2007; Volume 16, pp. 463–476, doi:10.1007/978-3-540-73729-2\_44. [\[CrossRef\]](#)
2. Shiue, Y.T.; Huang, H.H.; Chen, H. Correcting Chinese Word Usage Errors for Learning Chinese as a Second Language. In Proceedings of the COLING, Santa Fe, NM, USA, 20–26 August 2018.
3. Cheng, S.M.; Yu, C.H.; Chen, H.H. Chinese Word Ordering Errors Detection and Correction for Non-Native Chinese Language Learners. In Proceedings of the COLING 2014, the 25th International Conference on Computational Linguistics, Dublin, Ireland, 23–29 August 2014; Technical Papers; Dublin City University and Association for Computational Linguistics: Dublin, Ireland, 2014; pp. 279–289.
4. Eason, G.; Noble, B.; Sneddon, I. On certain integrals of Lipschitz-Hankel type involving products of Bessel functions. *Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Sci.* **1955**, *247*, 529–551.
5. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; doi:10.3115/1073083.1073135. [\[CrossRef\]](#)
6. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2016**, arXiv:1409.0473.
7. Ge, T.; Zhang, X.; Wei, F.; Zhou, M. Automatic Grammatical Error Correction for Sequence-to-sequence Text Generation: An Empirical Study. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; Association for Computational Linguistics: Florence, Italy, 2019; pp. 6059–6064, [\[CrossRef\]](#)

8. Schmalz, A.; Kim, Y.; Rush, A.M.; Shieber, S. Sentence-Level Grammatical Error Identification as Sequence-to-Sequence Correction. In Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications, San Diego, CA, USA, 16 June 2016; Association for Computational Linguistics: San Diego, CA, USA, 2016; pp. 242–251, doi:10.18653/v1/W16-0528. [CrossRef]
9. Li, S.; Zhao, J.; Shi, G.; Tan, Y.; Xu, H.; Chen, G.; Lan, H.; Lin, Z. Chinese Grammatical Error Correction Based on Convolutional Sequence to Sequence Model. *IEEE Access* **2019**, *7*, 72905–72913. doi:10.1109/ACCESS.2019.2917631. [CrossRef]
10. Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *arXiv* **2018**, arXiv:1808.03314.
11. Recurrent Neural Network. Available online: [https://www.cs.toronto.edu/~tingwuwang/rnn\\_tutorial.pdf](https://www.cs.toronto.edu/~tingwuwang/rnn_tutorial.pdf) (accessed on 20 November 2020).
12. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. *arXiv* **2014**, arXiv:1409.3215.
13. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. doi:10.1162/neco.1997.9.8.1735. [CrossRef] [PubMed]
14. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
15. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
16. Loper, E.; Bird, S. NLTK: The Natural Language Toolkit. *arXiv* **2002**, arXiv:0205028.
17. Zhao, Y.; Jiang, N.; Sun, W.; Wan, X. Overview of the NLPCC 2018 Shared Task: Grammatical Error Correction. In Proceedings of the 7th CCF International Conference, NLPCC 2018, Hohhot, China, 26–30 August 2018; pp. 439–445, doi:10.1007/978-3-319-99501-4\_41. [CrossRef]
18. Chen, T.; Li, M.; Li, Y.; Lin, M.; Wang, N.; Wang, M.; Xiao, T.; Xu, B.; Zhang, C.; Zhang, Z. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *arXiv* **2015**, arXiv:1512.01274.
19. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
20. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv* **2014**, arXiv:1406.1078.

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).