Hindawi
Security and Communication Networks
Volume 2023, Article ID 9857543, 1 page
https://doi.org/10.1155/2023/9857543

Retraction

Retracted: The Construction of Machine Translation Model and Its Application in English Grammar Error Detection

Security and Communication Networks

Received 26 December 2023; Accepted 26 December 2023; Published 29 December 2023

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] F. Long, "The Construction of Machine Translation Model and Its Application in English Grammar Error Detection," *Security and Communication Networks*, vol. 2021, Article ID 2731914, 11 pages, 2021.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 2731914, 11 pages
https://doi.org/10.1155/2021/2731914

Research Article

The Construction of Machine Translation Model and Its Application in English Grammar Error Detection

Fei Long [1,2]

[1]*Foreign Language School, Harbin University of Commerce, Harbin, Heilongjiang 150028, China*
[2]*Beijing Foreign Studies University, Artificial Intelligence and Human Languages Lab, Beijing 10089, China*

Correspondence should be addressed to Fei Long; 102622@hrbcu.edu.cn

Received 20 October 2021; Revised 16 November 2021; Accepted 1 December 2021; Published 18 December 2021

Academic Editor: Jian Su

In order to solve the problems of low accuracy, recall rate, and *F*1 value of traditional English grammar error detection methods, a new machine translation model is constructed and applied to English grammar error detection. In the encoder-decoder framework, the machine translation model is constructed through the steps of word vector generation, encoder language model construction, decoder language model construction, word alignment, output module, and so on. On this basis, the machine translation model is trained to detect English grammatical errors through dependency analysis and alternative word generation. Experimental results show that the accuracy, recall rate, and *F*1 value of the proposed method are higher than those of the experimental comparison method for detecting English grammatical errors such as articles, prepositions, nouns, verbs, and subject-verb agreement, indicating that the proposed method is of high practical value.

1. Introduction

As the most basic, direct, and convenient expression tool of human thoughts and feelings, natural language is always filled in every corner of human society. From the first cry after birth, people try to use language to express their feelings and intentions. With the advent of the information age, the forms of communication and communication in which people use natural language increasingly reflect its diversity, flexibility and universality [1]. In the era of globalization, as countries and nations are increasingly connected, translation between different languages has naturally become an urgent need. From basic overseas travel, literature translation to international business such as global e-commerce, financial services, international trade, etc., it is inevitable to encounter a problem of cross-language translation. Therefore, efficient translation between different languages in different fields has become a common demand of contemporary people. Translation is the most direct and effective way to solve communication barriers between natural languages [2]. Traditional translation work is done by professional translators, who can translate natural languages to each other through linguistic knowledge to help people realize information communication. With the maturity of computer technology and the continuous progress of natural language processing technology, it has become an inevitable trend to use machine translation system to help people get foreign information quickly instead of human translation. In this case, the large amount of data to be translated drives people to turn to machine translation systems for efficient translation. In addition, there are a lot of grammatical errors in the translation results due to the flood of machine translation software and backward technology at the present stage. Therefore, it is necessary to build an excellent machine translation model and detect its English grammatical errors so as to promote the further development of English translation [3].

To solve this problem, reference [4] proposed a syntax error detection method based on deep learning. According to the characteristics of grammar error diagnosis and the problems of existing grammar error diagnosis methods, a model combining BI-LSM-ATT and CRF based on attention mechanism is proposed to study grammar error diagnosis. The model uses jieba word segmentation technology to

preprocess data such as word segmentation and part of speech tagging and uses skip-Gram model to obtain word vector representation, which is used as the word embedding layer of the BI-LSTM-ATT model. The long-distance information in two directions is provided to the CRF model for sequence tagging, so as to obtain grammar error detection results. Reference [5] proposed a syntax error detection method based on data augmentation and replication. A copy mechanism is introduced into the self-attention model, and a new C-Transformer model is constructed. A text syntax error correction model from incorrect text sequence is constructed to correct text sequence. On the basis of public dataset, we use sequence to sequence learning to learn different forms of incorrect text from correct text and design a method of error text filtering based on smoothness, semantic and syntactic measures to obtain the result of grammar error detection. Reference [6] proposed a corpus-based English grammar error detection method. Compile corpus, and preprocess data, eliminate data noise, so as to obtain high-quality data. The preprocessed data are automatically segmented and marked with part of speech, and then the N-gram model is established to detect English grammar errors.

However, the above methods have low accuracy, recall rate, and $F1$ value. In order to solve these problems, this paper constructed a new machine translation model and applied it to English grammar error detection. The practical application effect of the model was verified through experiments.

Nowadays, the cross-language text processing mainly focuses on two directions, obtaining bilingual related links from parallel corpora or introducing feature information related to downstream tasks to strengthen the performance of text processing. The former needs a lot of annotation data, while the latter is suitable for specific downstream tasks, such as news topic classification and event detection, but cannot be extended to general tasks.

In this paper, we proposed a novel machine translation model. The organization of the paper is introduced as follows. In section 2, we give a detail introduction of the model, method, and technologies used in the model. In Section 3, we illustrate the English grammar error detection method proposed by us in detail. Section 4 shows the experiment of the paper and the corresponding results and we give the conclusion in Section 5.

2. Machine Translation Model Construction

In the framework of encoder and decoder, encoder, word alignment, and decoder are introduced, respectively. Decoders and encoders correspond to the processing process of source language and target language, respectively, and word alignment establishes the corresponding relationship between source language and target language [7]. For decoder and encoder, the common word vector generation and language model are introduced, respectively. And also in our model, we use the attention mechanism to implement the capture the corresponding attention weight relationship between multiple languages. However, unlike the encoder, the decoder ultimately needs to generate translation output, so an output module is added. The frame structure of the whole model is shown in Figure 1.

The whole process is implemented under a large encoder-decoder framework. The word alignment module establishes the connection between them. The encoder has two parts: word vector generation and language model. Besides, the decoder also has an output module. The entire translation process starts from the encoder side, the encoder word vector generation module inputs the vector sequence and generates the corresponding source language words, and then the language model encoder generates the language model sequence. Finally, the output of the source-side language model is synthesized. The word vector module is the link between the encoder and the decoder. Before the decoder conducts operations to generate outputs, the word alignment calculates the correlation degree of each output step and each input of the encoder and carries out weighted sum, so as to further serve the language model part of the decoder [8]. Once the word alignment calculation is complete, the decoder side operation can be performed to generate the translation output. The decoder calculates the current moment language model according to the output of the word alignment module, the previous moment language model, and the word vector generation module and further calculates the translation result of the current moment based on this output. Finally, the model operation of word vector generation is carried out on the obtained translation output [9] to prepare for the output at the next moment, and English machine translation is realized through the output model.

2.1. Encoder-Decoder Framework. Since 2013, end-to-end framework pattern has gradually emerged in the field of neural machine translation, which realizes direct conversion from source language sequence to target language sequence. End-to-end neural machine translation completely eliminates the process of manual translation process design. For example, there are no separate processing processes such as word alignment and syntax tree, but a joint neural network is established to complete the processing of natural language tasks under the framework of encoder-decoder [10, 11]. The framework maps a given source language sentence to a dense and continuous vector through an encoder, and then a target language sentence from the source language vector through a decoder. This is a very bold new structure. It replaces the linear model in machine translation with nonlinear model, and there is no hidden structure pipeline, but a single complex neural network. Time series network models can also capture historical information without site. Figure 2 is the architecture of the encoder-decoder framework, and Figure 3 is the encoder-decoder frame diagram.

As show in Figure 2, there are 3 different parts in the model: the encoder, the intermediate (encoder) vector, and decoder. For the encoder, it is a stack of several recurrent

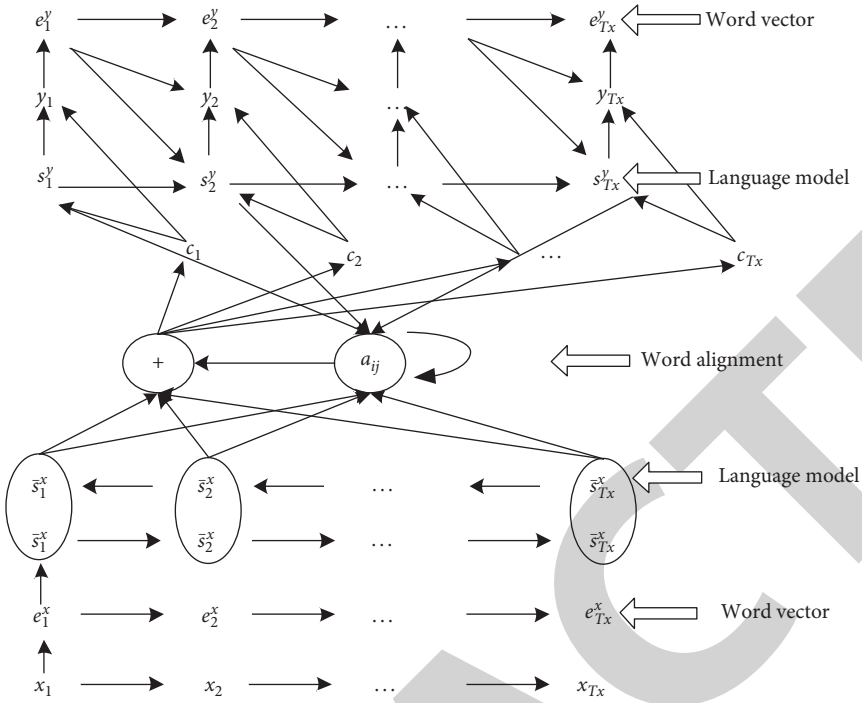

Figure 1: Overall architecture of the translation model.

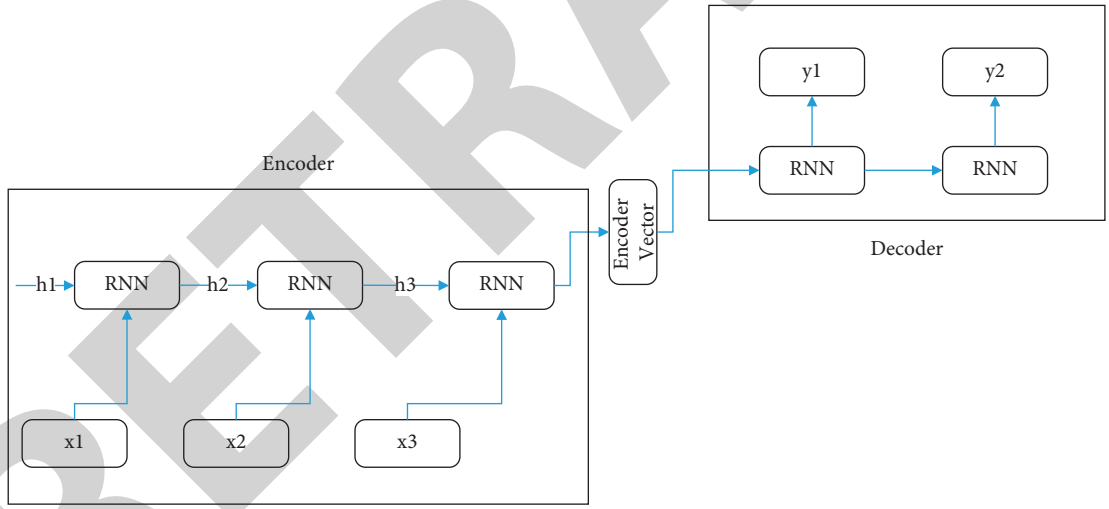

Figure 2: The architecture of encoder-decoder.

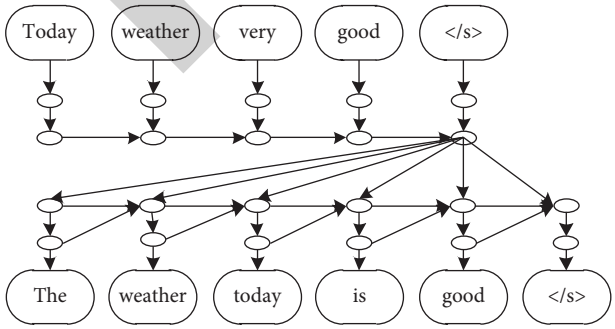

Figure 3: Frame diagram of encoder-decoder.

units (LSTM or GRU cells for better performance) where each accepts a single element of the input sequence, collects information for that element, and propagates it forward. The hidden states depend not only on the previous hidden state, but also on the input vector of the previous layer. For the encode vector, it is the final hidden state produced from the encoder part of the model. This vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions and it also acts as the initial hidden state of the decoder part of the model. For the decoder, it is a stack of several recurrent units where each unit predicts an output at a special time step. Each recurrent

unit accepts a hidden state from the previous unit and produces and output as well as its own hidden state.

As shown in Figure 3, assuming that a Chinese source statement "today weather very good" is input, the model first generates word vectors for each word and then generates sentence vectors in sequence through the time series network model. The last symbol represents the sentence terminator. The part of the source language side that generates a vector representation of a dense and continuous sentence from the input corresponds to an encoder bias [12]. The target language end corresponds to the source language end, which has a similar structure with the encoder end and adopts the same RNN series network, but the purpose of the decoder end is to generate translation output. After the output statement "The weather today is good</s>," the decoding part is complete. Since the model is sequential, the output of the current moment depends on the output information of the previous moment.

2.2. Word Vector Generation. Most of the distributed representation-based methods are word-level. Of course, the mainstream natural language processing first uses various word dividers to divide a sentence into words. However, this topic also involves a training method of character data. In order to optimize the training of such data, a word vector generation method based on recursive neural network (RNN) is proposed.

RNN computes a variable length input vector x to obtain hidden layer h and y vectors for alternative outputs. Every time t passes, the hidden layer h_t of RNN will be updated iteratively according to the following formula [13]:

$$h_t = f(h_{t-1}, x_t), \tag{1}$$

where f represents an activation function, usually nonlinear, which can be a simple SigmoID function [14] or a complex long short-term memory network (LSTM) unit. Then, after repeated training of the next possible symbol in the sequence, RNN can automatically learn the corresponding probability distribution [15]. In this way, the output at t will be a conditional distribution $p(x_t|x_{t-1}, \ldots, x_1)$, as in formula (2), a one-hot polynomial distribution can use Softmax as the activation function.

$$p(x_t \mid x_{t-1}, \ldots, x_1) = \frac{\exp(w_j h_t)}{\sum_{j'=1}^{K} \exp(w_{j'} h_t)}, \tag{2}$$

where $w_j\,(j = 1, \ldots, K)$ is a row in the weight matrix W. By combining these probabilities, we can calculate the probability of the occurrence of a sequence x using the following formula:

$$p(x) = \prod_{t=1}^{T} p(x_t \mid x_{t-1}, x_1). \tag{3}$$

Then, the task of obtaining word vector can also be completed by extending the basic idea of RNN. h_T, the hidden layer of the last step, at time T, is already a real vector, but it is not a word vector, and it is a vector representation of a sentence. If we save an additional download of h_t of each time t and clear it as input for the next hidden layer, we get a chronological sequence of word vectors $(h_1, \ldots, h_T)$. Of course, on closer inspection, this is equivalent to $h_t = f(x_t)$; in other words, a nonlinear transformation based on the one-Hot code. However, the general idea of this scheme is feasible. RNN is mainly used to generate vector representations, but it is only necessary to design how to selectively set h_t to zero at the appropriate time.

From a natural language perspective, clear zero can be understood as a cut word, which happens to be required for a model that takes character data as input. Then, a switch can be added to the neural network to output the segmented word vector and clear the information of the last word in time. The word vector generating unit at each time t, which is specific to an input sequence, will be expanded in the direction of the chronological order. Among them, h is the hidden layer unit of recursive neural network [16], and its basic recursion mode is the same as that of RNN. w is the switch used to control word segmentation, e is the output of word vector, and it is also subject to switch w, and this switch w will only and must clear h or e at the same time.

The switch layer w_t should also be related to $f(h_{t-1}, w_{i-1}, x_t)$, and the corresponding word vector generation method is shown as follows:

$$h_i = \tanh(W_h x_i + U_h h_{i-1}), \tag{4}$$

$$\overline{w}_i = \sigma(W_w x_i + U_w h_{i-1} + V_w w_{i-1}), \tag{5}$$

$$w_i = \begin{cases} 0, & \text{if } w_{i,1} \geq w_{i,2}, \\ 1, & \text{if } w_{i,1} < w_{i,2}, \end{cases} \tag{6}$$

$$e_y^{i-1} = w_i h_i, \tag{7}$$

$$h_i = (1 - w_i) h_i, \tag{8}$$

where $x = (x_1, \ldots, x_T)$, $x_t \in R^{K_x}$, K_x is the dictionary size of the source language, and T_x is the sentence length of the source language and is generally not equal to different sentences, σ is the sigmoid activation function, $W_h \in R^{m \times K_x}$, $U_h \in R^{m \times m}$, $W_w \in R^{2 \times K}$, $U_w \in R^{2 \times m}$ are the weight matrix, and m is the short vector dimension after word embedding. h_i in formula (4) is the output of the original hidden layer, and $h_i = \overrightarrow{0}$. w_i in formula (5) can be understood as a switch to determine whether h_i at time i is output, similar to the effect of dynamic word segmentation. To ensure that e_{i-1}^{y} does not contain information from the previous word, formula (8) is a further update operation for h_i.

2.3. Encoder Language Model. NNLM was first systematically proposed and studied in depth by Bengio. The purpose of this algorithm is to predict the current word using the first $n-1$ lyrics of the current word. It should be noted that the purpose of splicing $n-1$ vectors is to input the first $n-1$ words of the current word as a vector. Of course, you can choose whether to do so or not. The advantage of this is that it is simple and straightforward, or you can choose to combine it into a matrix, even with convolutional neural

networks and recursive neural networks. The dashed line input layer is directly connected to the output layer, so that the original input and the information of the hidden layer transformed by one layer are simultaneously used as the input of the output layer, which is equivalent to extending the basic features. The language model generation unit is shown in Figure 4.

As shown in Figure 4, when we read a sentence, we process each word accumulating information up to the end of the text. A system accumulating information composed of similar units repeated over time is a recurrent neural network (RNN) in the deep learning field. In general, a text encoder turns text into a numeric representation. This task can be implemented in many different ways but what we mean by encoders are RNN encoders. Here, $x(i)$ denotes the input sentences and is formulated as a work vector, the hidden state vector $h(i)$ contains the sequence state before the current block, W_x are the weights between $x(i)$ and $h(i)$, W_h are the weights between $h(i+1)$ and $h(i)$, W_y are the weights between $h(i)$ and $o(i)$, and $o(i)$ is the output vector which is not always produced by each block.

Although two control gates and one output gate are simply added, RNN does not have this function by comparison. The following is the specific language model generation formula.

$$\overrightarrow{z}_i = \sigma\left(\overrightarrow{W}_z e_i^x + \overrightarrow{U}_z \overrightarrow{s}_{i-1}\right), \tag{9}$$

$$\overrightarrow{r}_i = \sigma\left(\overrightarrow{W}_r e_i^x + \overrightarrow{U}_r \overrightarrow{s}_{i-1}\right), \tag{10}$$

$$\overrightarrow{g}_i' = \tanh\left(\overrightarrow{W}_g e_i^x + \overrightarrow{U}_g (\overrightarrow{r}_i \circ \overrightarrow{g}_{i-1})\right), \tag{11}$$

$$\overrightarrow{g}_i = \left(1 - \overrightarrow{z}_i\right) \circ \overrightarrow{g}_{i-1} + \overrightarrow{z}_i \circ \overrightarrow{g}_i, \tag{12}$$

$$\overrightarrow{s}_i = \sigma\left(\overrightarrow{W}_s e_i^x + \overrightarrow{U}_s \overrightarrow{s}_{i-1}\right) \circ \tanh(\overrightarrow{g}_i). \tag{13}$$

Among them, $e_i^x \in R^m$ is the m-dimensional word embedding vector embedded into vector module, and $\overrightarrow{W}_z, \overrightarrow{W}_r, \overrightarrow{W}_g \in R^{n \times m}$ weight matrix, and $g_0 = 0, S_0 = 0$, formula (9) in z_i is updated, formula (10) in r_i is reset, the former allows each hidden unit to maintain their previous activated state, while the latter controls what information and how much information need to reset from the previous state. s_i in formula (13) is the output gate, which is an improvement on the GRU model. At different times t, each hidden layer unit has its own reset, update, and output gates that capture dependencies at different time scales. The reset gate is activated frequently when the hidden layer unit attempts to capture short-term memory; the same is true of renewal gates, which tend to capture long-term memories.

2.4. Decoder Language Model. Since it is a machine translation system, there are two relative natural languages, in other words, two relatively independent language models. Through the improved bidirectional GRU in the encoder, the hidden layer sequence s is obtained. The same decoding module s also has a similar GRU structure. There is no need to add an output gate because this function has been reflected in the output module. But if this is all, there will be no connection between encoder and decoder, and there will be no translation process. Therefore, in the calculation process of decoder module s', c containing encoding module information is also added. Here, c can be temporarily regarded as all or part of the source language information, and the specific definition of c will be given in the word alignment model. Therefore, the specific generation algorithm of the language model in the decoder is as follows:

$$\begin{aligned} z_i &= \sigma\left(W_z e_{i-1}^y + U_z s_{i-1}' + C_z c_i\right), \\ r_i &= \sigma\left(W_r e_{i-1}^y + U_r s_{i-1}' + C_r c_i\right), \\ \overline{s}_i' &= \tanh\left(W_s e_{i-1}^y + U_s\left[r_i \circ s_{i-1}'\right] + C_s c_i\right), \\ s_i' &= \left(1 - z_i\right) \circ s_{i-1}' + z_i \circ s_i', \end{aligned} \tag{14}$$

where $e_i^y \in R^m$ is the m-dimensional word embedding vector of the target language, that is, the result generated by the language model; $\circ$ is the dot product calculation; $W_z, W_r, W_s \in R^{n \times m}$, $U_z, U_r, U_s \in R^{n \times n}$, and $C_z, C_r, C_s \in R^{n \times n_l}$ are the weight matrix; $s_0' = \tanh(V_s \overrightarrow{s}_1)$, and the decoder part does not need to calculate the bidirectional language model because it further serves the output module, so as to output the translated target language sequence in turn.

2.5. Word Alignment Based on Global Information. Word alignment is the most fundamental problem in neural machine translation. Most neural machine translation problems are based on encoder-decoder framework, encoding and decoding corresponding to the source language and target language, respectively. But if the two parts are clearly separated in this way, then the source language and the target language will no longer have any connection, and there will be no translation problem. Word alignment is to establish the connection between the source language end and the target language end, learn the corresponding relationship between the source language and the target language from the parallel corpus, and further learn the translation rules on this basis, so as to complete the training of neural machine translation model. Therefore, word alignment plays an important role in the final result of neural machine translation system. Based on the attention mechanism, this paper uses a global word alignment method by integrating historical information.

Because the two-way loop network is used in the language model design, language translation is uncertain. The words each decoder outputs now depend on a weighted combination of all the input states, not just the last state. Weights determine how much each input state contributes to the output state. Therefore, if the weight is large, it means that the decoder will pay more attention to the corresponding part in the original sentence when generating the current word of the translation. The result of the weight sum is usually normalized to 1 (which is a distribution of the input states).

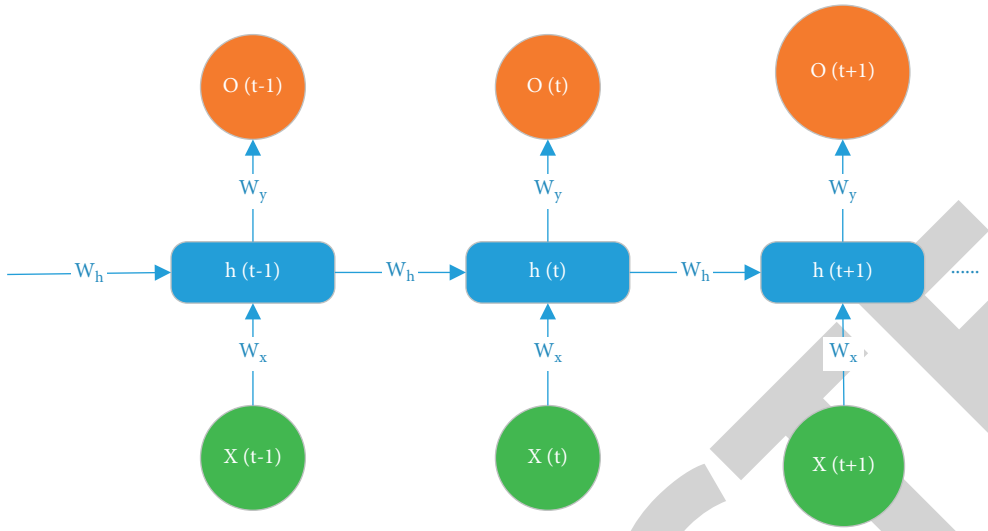

Figure 4: Language model generation unit.

Weights are usually calculated from the output of the language model on the source side and the information of the hidden layer at the last moment on the target side. However, the weight is relative, so its calculation should not only be affected by the above two factors, but also related to the historical information of the weight. The corresponding calculation methods are shown in formulas (15) to (20).

$$e_{ij} = V_a^T \tanh\left(W_a s_{i-1}^y + U_a s_j^x\right), \tag{15}$$

where V_a, W_a, and U_a represent the weight matrix; s_{i-1}^y represents the output of the language model at the target language side at the previous moment; and s_j^x represents the output of the source language-side language model:

$$b_{ij}^h = \sum_{i=1}^{t-1} \exp\left(e_{ij}\right), \tag{16}$$

$$b_{ij} = \frac{\exp\left(e_{ij}\right)}{b_{ij}^h}, \tag{17}$$

$$\alpha_{ij} = \frac{b_{ij}}{\sum_{k=1}^{T_k} b_{ik}}, \tag{18}$$

$$\alpha_{ij} = \frac{b_{ij}}{\sum_{k=1}^{T_k} b_{ik}}, \tag{19}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} s_j, \tag{20}$$

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E\left((X-\mu_X)(Y-\mu_Y)\right)}{\sigma_X \sigma_Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{N\sum X^2 - (\sum X)^2}\sqrt{N\sum Y^2 - (\sum Y)^2}}. \tag{21}$$

In the calculation of attention, it is necessary to calculate the degree of correlation between each source language word and each target language word.

During the training phase, the model's parameter, such as the weights and the basis, is initialized according to a standard Gaussian distribution, which means that the mean value of the weights is 0 and the standard deviation of the weights is 1. The basis of the model is initialized to be all zeros.

2.6. Output Module. As for decoders, we introduced the common word vector generation and language model of encoder-decoders in the previous part. Decoders differ from encoders in that a final output is necessary for the decoder to achieve supervised learning. However, the previous section alone does not produce translation output. Therefore, the model finally adds an output module for supervised learning, so that effective translation results can be obtained in the end.

In order to obtain the translation result, the result e_{i-1}^y of the word vector module at the decoder side, the result e_i^y generated by the language model, and the source language information c_i generated by word alignment should be fused at every moment before the generation of terminators. The calculation method of the output module is shown in formulas (21) to (23).

$$\bar{t}_i = \sigma\left(W_i e_{i-1}^y + U_i s_i^y + C_i c_i\right), \tag{22}$$

$$t_i = \max\left\{t_{t,2j-1}, t_{t,2j}\right\}, \quad j = 1, 2, \ldots, l, \tag{23}$$

$$p\left(y_i \mid e_{i-1}^y, s_i^y, c_i\right) = y_i^T \operatorname{softmax}\left(V_p t_i\right), \tag{24}$$

where W_t, U_t, C_t, and V_p represent the weight matrix, and the dimensions are $2l * m$, $2l * n$, $2l * n\prime$, and $k_y * l$, respectively. max represents nonlinear activation function; soft max represents the activation function.

2.7. Attention Mechanism. Attention mechanism is a mechanism used in encoder-decoder structure [17]. Now it has been used in a variety of tasks, including machine translation, image description, and text summarization. It is no longer limited to encoder-decoder structure [18]. A variety of variant attention structures are applied in various tasks. In general, the application of attention mechanism includes the following:

(1) Allowing the decoder to pay attention to the information it needs in multiple vectors in the sequence is the use of the traditional attention mechanism. Because the encoder multistep output is used instead of a single certain length vector of the corresponding step, more information is retained.

(2) Acting on the encoder to solve the representation problem (for example, encoding vector is used as the input of other models), self-attention [19] is generally used.

3. English Grammar Error Detection

3.1. Model Training. When training the model, first parse each sentence in the corpus, generate short sentences according to the method of generating clauses in the machine translation model constructed above, and save all clauses at all levels into a new corpus. This parsed corpus is then processed by a machine translation model generation tool to generate statistical information. Finally, the machine translation model is stored in a database for searching.

The training process of machine translation model is as follows. Firstly, every sentence in the corpus is analyzed, and hierarchical clauses are generated and put into hierarchical clause set. The probability information of the clause set, namely, the tuple probability table, is generated by the machine translation model tuple probability generation tool and stored in the database for query. The corpus used in this paper is the whole corpus of Yahoo N-gram corpus, the corpus dependency analysis tool uses Stanford Parser, the machine translation model tuple probability generation tool uses Berkeley LM, and the smoothing method uses Kneser–Ney smoothing. The machine translation model has a high requirement for memory during the training process. Low memory settings will take a long time and are prone to error, so more memory will be allocated in general. In this article, the machine translation model training tool uses Berkeley LM to generate ternary models.

3.2. Dependency Analysis and Generation of Alternative Words. Since the machine translation model in this paper is a language model generated by dependency analysis, a sentence to be corrected needs to go through dependency analysis to generate hierarchical structure and then use statistical information generated by the corpus to correct grammatical errors. Errors in the use of words can generally be divided into two types: spelling errors and correct spelling but not in accordance with the context of the true word errors. Spelling errors can be searched using Lucene's SpellChecker dictionary feature. That is, typing in a word returns a set of edit-close words in a preprepared dictionary, which can consist of tuples of words from the Yahoo N-gram corpus that Berkeley LM has trained.

True word errors need to be considered separately. In the determiner errors, this paper deals with the use of "the," "a," and "an" separately. When these three words appear, they are alternatives to each other. The alternative of prepositions is similar. This paper only deals with the most common prepositions, that is, "in," "on," "to," "of," and "for" are alternatives to each other.

For obtaining other alternative words, this article uses the functions provided in Freeling. Freeling can perform word state analysis on a given word and extract related words with the same stem as generated words. These related words are used as alternatives in this paper. Such errors as verb morphology and noun singular and plural and subject-verb consistency can be provided by this method for English grammatical error detection.

3.3. Error Correction Decoding Algorithm. Consider sentence s with only two levels of nodes. Let the sentence length be n, i.e., only dominant words) and 33 subordinate words, and there are no subordinate words under these $n-1$, $n-1$ subordinate words. There is only one hierarchical clause in this sentence after dependency analysis, and the order of the words in the hierarchical clause is consistent with the original sentence. Using the ternary language model in this paper, the probability calculation method of sentences is as follows:

$$p(s) = \prod_{i=1}^{n} p(w_i | w_{i-1}, w_{i-2}). \tag{25}$$

Here, we use formula (25) to calculate the probability of the sentence s, which depends on the previous weights' information of the model, and the probability of the sentence is also assumed conditionally independent requirements.

Assuming that the root of the hierarchical clause is a certain word, but there are multiple alternative leaves, the decoding sequence with the highest probability can be obtained through the Viterbi algorithm. The maximum probability of defining this decoding sequence is the score of the hierarchical clause with the root of n, that is, $\text{subScore}(n)$.

To unify the process, more generally the leaf is also defined as the root of a tree, and if this leaf is set to w, the trees that define its alternatives all score subScore.

Generally speaking, a sentence may have multiple levels after parsing, so the score defining this tree can be recursively defined as: the score of the first-level clause multiplied by the score of each subtree formed by nonroot nodes in the first-level clause. Let the word order of the first-level subtree be listed as $\text{subSeq} = c_1, \ldots, c_k r c_{k-1}, \ldots, c_{n-1}$, where r is the root of the whole sentence and c is the other $n-1$ nodes dependent on the head node, namely,

$$p(s) = \text{subScore}(r) = p(\text{subSeq}) \prod_{i=1}^{n-1} \text{subScore}(c_i). \tag{26}$$

Combined with the above analysis results, the English grammar error detection process is mainly composed of three processes: dependency analysis to generate hierarchical clauses, generation of alternative words, and finally language model decoding to generate English grammar error detection results. The overall process is shown in Figure 5.

The program starts by reading a statement from the text, dividing the statement into words, then obtaining the dependencies of each word, and recombining to generate hierarchical clauses. Then the candidate word tool is used to obtain the candidate word for each word, and the decoding process of the machine translation model begins. The decoding process starts with the topmost clause and, using the probabilities calculated by the machine translation model, computes and retains the maximum probability path for a series of words in their current position. Because each word may have a modification part, this process needs to carry out recursive calculation, according to the result of recursive calculation to achieve English grammar error detection.

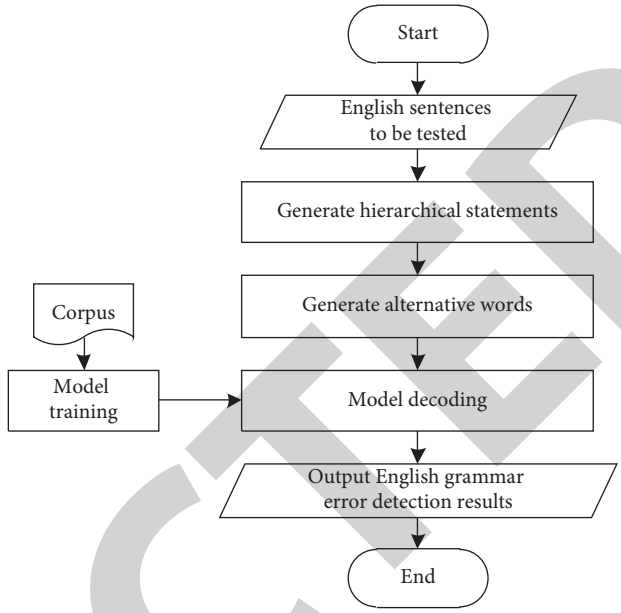

Figure 5: Flowchart of English grammar error detection.

4. Experimental Results and Analysis

4.1. Experimental Design

4.1.1. Experimental Data. Experimental data came from GEC evaluation task of CONLL2013. Statistical results of training data and test data are shown in Table 1. The test data were divided into two parts with the same number of sentences: long and short.

4.1.2. Corpus. This corpus is from Yahoo N-gram (version 2.0; $n = 1, 2, 3, 4, 5$). The N-gram corpus is based on news from 12,000 news websites, with 14.6 million articles, 126 million sentences, and 3.4 billion words. This corpus does not contain the original corpus article itself, but only contains N-gram fragments that appear at least twice. Due to the large scale of N-gram language model, in order to improve the retrieval efficiency, this paper uses the open-source search engine Apache Solr to build index and search services for it. The preprocessing procedure is illustrated in Figure 6.

4.1.3. Evaluation Indicators. The evaluation of experimental results was based on CONLL2013 evaluation standard, and accuracy P, recall R, and $F1$ values were adopted, which were defined as follows:

$$P = \frac{N_{\text{correct}}}{N_{\text{predicted}}},$$

$$R = \frac{N_{\text{correct}}}{N_{\text{target}}}, \tag{27}$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R},$$

where N_{correct}, $N_{\text{predicted}}$, and N_{target}, respectively, refer to the number of correct corrections, the number of corrected errors, and the number of errors that should have existed.

Table 1: Details of training data and test data.

Wrong type	Training set		Test set	
	Amount	Proportion	Amount	Proportion
Articles	6658	14.8	690	19.9
Prepositions	2404	5.3	312	9.0
Noun	3779	8.4	396	11.4
Subject-predicate agreement	1453	3.2	122	3.5
Verb forms	1527	3.4	124	3.6
Total error data	15821	35.1	1644	47.4
Total data	45106	—	3470	—

The syntax error detection method based on deep learning, the syntax error detection method based on data augmentation and replication, and the proposed method are used as experimental comparison methods to verify the application effects of different methods by comparing different indicators.

4.2. Analysis of Experimental Results

4.2.1. Short Test Set. The English grammar error detection results of the syntax error detection method based on deep learning, the syntax error detection method based on data augmentation and replication, and the method in this paper based on the short test set are shown in Tables 2 to 4.

For article errors, the accuracy, recall rate, and F1 value of the proposed method are 2.34% and 7.44% higher than those based on deep learning, 0.71% and 3.71% higher than those based on data augments and replication, respectively, and the $F1$ value is 1.3% and 6.84% higher.

For preposition errors, the accuracy and $F1$ values of the proposed method are both higher than those of deep learning-based grammar error detection method and data augmentation and replication method, which are 3.26%

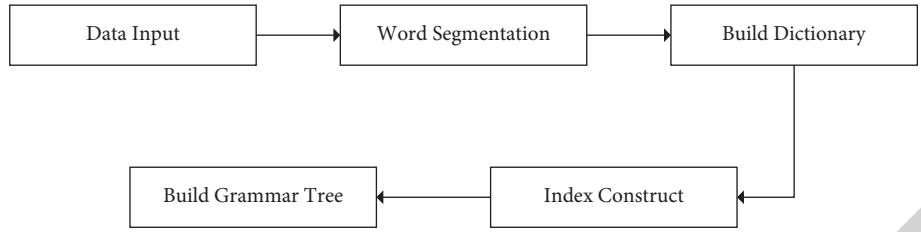

Figure 6: The preprocessing procedure of the corpus.

Table 2: Comparison results of recall rates.

Wrong type	Syntax error detection method based on deep learning	Syntax error detection method based on data augmentation and replication	Model without attention mechanism	Paper method
Articles	0.3111	0.2601	0.3021	0.3345
Prepositions	0.0574	0.0452	0.0613	0.0778
Noun	0.2757	0.2997	0.3101	0.3213
Verb	0.1599	0.2302	0.2258	0.2704
Subject-predicate agreement	0.1402	0.2502	0.2107	0.3014

Table 3: Comparison results of accuracy rates.

Wrong type	Syntax error detection method based on deep learning	Syntax error detection method based on data augmentation and replication	Model without attention mechanism	Paper method
Articles	0.5032	0.4732	0.4512	0.5103
Prepositions	0.1679	0.1510	0.1423	0.1677
Noun	0.5268	0.4088	0.5112	0.5288
Verb	0.2595	0.2201	0.2498	0.2813
Subject-predicate agreement	0.2295	0.2401	0.2765	0.3122

Table 4: Comparison results of $F1$ values.

Wrong type	Syntax error detection method based on deep learning	Syntax error detection method based on data augmentation and replication	Model without attention mechanism	Paper method
Articles	0.3910	0.3357	0.3745	0.4041
Prepositions	0.0852	0.0730	0.1005	0.1062
Noun	0.3401	0.3458	0.3849	0.4131
Verb	0.1985	0.2250	0.2451	0.2757
Subject-predicate agreement	0.1749	0.2450	0.2695	0.3067

and 2.04%, and 2.1% and 3.32%, respectively, and the recall rate is 0.02% lower than that of deep learning-based grammar error detection method. Syntax error detection method based on data enhancement and replication reached 1.67%.

For noun errors, the accuracy, recall rate, and $F1$ value of the proposed method are 4.56% and 2.16% higher than those based on deep learning, 2% and 12% higher than those based on data enhancement and replication, and 7.3% and 6.73% higher than those based on data enhancement and replication, respectively.

For verb errors, the accuracy, recall rate, and $F1$ value of the proposed method are 11.05% and 4.02%, 2.18% and 6.12%, and 7.72% and 5.07% higher than those based on deep learning and data augments and replication, respectively.

For subject-verb-induced errors, the accuracy, recall rate, and $F1$ values of the proposed method are 16.12% and 5.12% higher than those of the deep learning-based method, 8.27% and 7.21% higher than those of the data augment-based method and replication method, and 13.18% and 6.7% higher than those of the deep learning-based method and data augment-based method, respectively.

4.2.2. Long Test Set. The English grammar error detection results of the syntax error detection method based on deep learning, the syntax error detection method based on data augmentation and replication, and the method in this paper based on the long test set are shown in Tables 5 to 7.

As shown in Tables 4–7, for article errors, the accuracy, recall rate, and $F1$ value of the proposed method are 7.18%

Table 5: Comparison results of recall rates.

Wrong type	Syntax error detection method based on deep learning	Syntax error detection method based on data augmentation and replication	Model without attention mechanism	Paper method
Articles	0.2697	0.2597	0.3321	0.3415
Prepositions	0.0472	0.0507	0.052	0.0781
Noun	0.2315	0.3321	0.2983	0.3516
Verb	0.1513	0.2521	0.2745	0.2913
Subject-predicate agreement	0.1102	0.2712	0.29	0.3219

Table 6: Comparison results of accuracy rates.

Wrong type	Syntax error detection method based on deep learning	Syntax error detection method based on data augmentation and replication	Model without attention mechanism	Paper method
Articles	0.4742	0.4751	0.5012	0.5134
Prepositions	0.1590	0.1608	0.1658	0.1727
Noun	0.4533	0.4601	0.5288	0.5318
Verb	0.2201	0.2595	0.2032	0.3123
Subject-predicate agreement	0.1995	0.2732	0.33	0.3312

Table 7: Comparison results of $F1$ values.

Wrong type	Syntax error detection method based on deep learning	Syntax error detection method based on data augmentation and replication	Model without attention mechanism	Paper method
Articles	0.3438	0.3356	0.4023	0.4102
Prepositions	0.0730	0.0847	0.105	0.1076
Noun	0.3064	0.3829	0.414	0.4233
Verb	0.1798	0.2553	0.298	0.3014
Subject-predicate agreement	0.1749	0.2722	0.3132	0.3265

and 8.18% higher than those of the grammar error detection method based on deep learning and the grammar error detection method based on data augmentation and replication, respectively, and the recall rate is 3.92% and 3.83% higher. $F1$ values were 6.64 and 7.46% higher.

For preposition errors, the accuracy, recall rate, and $F1$ value of the proposed method are higher than those of deep learning-based grammar error detection method and data augmentation and replication-based grammar error detection method, which are 3.09% and 2.74%, 1.37% and 1.19%, and 3.46% and 2.3%, respectively.

For noun errors, the accuracy, recall rate, and $F1$ value of the proposed method are 12.01% and 1.95% higher than those of the deep learning-based method, 7.85% and 7.17% higher than those of the method based on data augment and replication, and 11.69% and 4.0% higher than those of the method based on data augmentation and replication, respectively.

For verb errors, the accuracy, recall rate, and $F1$ values are higher than those of deep learning-based grammar error detection method and data augmentation and replication-based grammar error detection method, respectively, by 14% and 3.92%, 9.22% and 5.28%, and 12.6% and 4.61%.

For subject-predicate agreement errors, the accuracy, recall rate, and $F1$ value of the proposed method are 22.17% and 5.07%, 13.17% and 5.8%, and 15.16% and 5.43% higher than those based on deep learning and data augmentation and replication, respectively.

5. Conclusion

In the era of globalization, as countries and nations are increasingly connected, translation between different languages has naturally become an urgent need. Although human translation can achieve accurate translation, its corresponding human cost is unaffordable for many needs. Therefore, the importance of machine translation is self-evident. However, machine translation is far from fully automatic and high-quality translation and has a great space for development. Therefore, this paper designs a new machine translation model and applies it to English grammar error detection. Experimental results show that in the short test set and long test set, the accuracy, recall rate, and $F1$ value of the proposed method are higher than those of the experimental comparison method for detecting English grammatical errors such as articles, prepositions, nouns,

verbs, and subject-verb agreement, and error detection results show that the method of English grammar of higher quality has good practical application value.

Data Availability

The data used to support the findings of this study are available upon request to the author.

Conflicts of Interest

The author declares that he has no conflicts of interest.

Acknowledgments

This study was supported by the "Doctoral research initiation fund of Harbin University of Commerce" Computation-Oriented Causal Relation Recognition (Project no. 2019DS105).

References

[1] H. Lei and W. Shao, "Application of E-business English translation in multi lingual system of picking robot-based on ESP theory," *Journal of Agricultural Mechanization Research*, vol. 42, no. 2, pp. 234–237, 2020.

[2] L. Y. Tay, L. S. Tan, J. Y. Tan, and T. B. Aiyoob, "Validity and reliability of an English translation of the Teacher Metacognition Inventory (TMI) with mathematics teachers in Singapore," *Current Psychology*, vol. 141, no. 1-2, pp. 1–10, 2021.

[3] G. P. Jacobson, P. Erin, H. Kelsey, and R. A. Richard, "A factor analytic assessment of the English translation of the neuropsychological vertigo inventory (NVI) - ScienceDirect," *Journal of Otology*, vol. 15, no. 2, pp. 45–49, 2020.

[4] H. Wang, J. H. n, and H. C. Wang, "Research on Chinese grammar error diagnosis method based on deep learning," *Computer Technology and Development*, vol. 30, no. 11, pp. 75–79, 2020.

[5] Q. B. Wang and Y. Tan, "Chinese grammatical error correction method based on data augmentation and copy mechanism," *CAAI Transactions on Intelligent Systems*, vol. 15, no. 1, pp. 99–106, 2020.

[6] H. C. Wang and J. C. Zhou, "Research and implementation of Chinese grammar automatic error correction system," *Sci-Tech & Development of Enterprise*, vol. 46, no. 2, pp. 89–92, 2020.

[7] A. El-Kishky, A. Renduchintala, J. Cross, and F. J. Guzman, "XLEnt: mining a large cross-lingual entity dataset with lexical-semantic-phonetic word alignment," *Computer Science*, vol. 17, no. 10, pp. 1–12, 2021.

[8] K. Khysru, D. Jin, Y. Huang, H. Feng, and J. Dang, "A Tibetan language model that considers the relationship between suffixes and functional words," *IEEE Signal Processing Letters*, vol. 5, no. 99, pp. 1–10, 2021.

[9] G. Elahi and Y. H. Yang, "Online learnable keyframe extraction in videos and its application with semantic word vector in action recognition," *Pattern Recognition*, vol. 25, no. 2, Article ID 108273, 2021.

[10] P. Frank and E. Sasse, "MDS-107: MDS patients' needs from online discussion forums: an artificial intelligence and natural language processing analysis of 20,000 posts in US, UK, Canada, and China," *Clinical Lymphoma, Myeloma & Leukemia*, vol. 20, no. 1, pp. S313–S314, 2020.

[11] S. Saunders, G. Muniz-Terrera, S. Sheehan, C. W. Ritchie, and S. Luz, "A UK-wide study employing natural language processing to determine what matters to people about brain health to improve drug development: the electronic person-specific outcome measure (ePSOM) programme," *The Journal of Prevention of Alzheimer's Disease*, vol. 23, no. 1, pp. 1–9, 2021.

[12] K. V. Gouthaman and A. Mittal, "Reducing language biases in visual question answering with visually-grounded question encoder," *Computer Science*, vol. 13, no. 1, pp. 18–34, 2020.

[13] M. Qiao, S. Yan, X. Tang, and C. Xu, "Deep convolutional and LSTM recurrent neural networks for rolling bearing fault diagnosis under strong noises and variable loads," *IEEE Access*, vol. 4, no. 99, pp. 1–13, 2020.

[14] S. Amin, M. I. Uddin, S. Hassan et al., "Recurrent neural networks with TF-IDF embedding technique for detection and classification in tweets of dengue disease," *IEEE Access*, vol. 8, no. 1, Article ID 131522, 2020.

[15] R. R. Li and Y. Dai, "Short term wind speed prediction based on LSTM and time series analysis," *Computer Simulation*, vol. 37, no. 3, pp. 393–398, 2020.

[16] P. Priyanga, V. V. Pattankar, and S. Sridevi, "A hybrid recurrent neural network-logistic chaos-based whale optimization framework for heart disease prediction with electronic health records," *Computational Intelligence*, vol. 37, no. 12, pp. 1–10, 2020.

[17] Z.-L. Lu and B. A. Dosher, "External noise distinguishes attention mechanisms," *Vision Research*, vol. 38, no. 9, pp. 1183–1198, 1998.

[18] B. A. Sangeroki and T. W. Cenggoro, "A fast and accurate model of thoracic disease detection by integrating attention mechanism to a lightweight convolutional neural network," *Procedia Computer Science*, vol. 179, no. 11, pp. 112–118, 2021.

[19] L. Xiang, P. Wang, X. Yang, A. Hu, and H. Su, "Fault detection of wind turbine based on SCADA data analysis using CNN and LSTM with attention mechanism," *Measurement*, vol. 175, no. 8, Article ID 109094, 2021.