Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105
Volume **10**, Number **3**, August **2025**

Research on Chinese Semantic Role Annotation Based on Deep Learning

Chen Zeng[1,*], Tian-Lin Shen[2]

[1]Xinyang Agriculture and Forestry University, Xinyang 464000, P. R. China
18603768307@163.com

[2]Yana Royal Polytechnic University, Chiang Mai 50000, Thailand
yf3029@163.com

[*]Corresponding author: Chen Zeng

Received May 21, 2024, revised October 18, 2024, accepted January 3, 2025.

Abstract. *Internet information is growing explosively. How to extract important knowledge from large-scale data becomes particularly important. By analyzing and processing unstructured text, the knowledge expressed is presented in a structured form, thereby constructing a large-scale knowledge graph, enabling computers to more accurately understand the content that humans want to express and accelerating their processing capabilities. Event recognition and extraction, as an important goal in the field of natural language processing, play a crucial role in the construction of knowledge graphs. However, due to the high complexity of this issue, it is difficult to handle. Semantic role annotation, as an important core technology for constructing event graphs, can meet the needs of shallow semantic analysis. This article focuses on the improvement of Chinese semantic role annotation technology. On the basis of existing models, self attention mechanism and high-speed bidirectional long short-term memory technology are added to solve the long-term dependency problem in the original model. Multiple factors that affect the model are compared and tested to find the best practical parameters. The experimental results show that the performance of the new model proposed in this article can be significantly improved, and the generalization ability of the model is greatly enhanced.*
Keywords: Semantic role labeling; Fusion model; Self attention mechanism; High speed bidirectional long short-term memory

1. **Introduction.** Semantic Role Labeling (SRL), as an important component of event recognition and extraction tasks, belongs to a sentence level shallow semantic analysis technique [1]. The goal of this task is to identify other parts of the sentence that are related to the predicate, based on the predicate in the sentence. We call them arguments, and then assign these arguments to the correct role categories. According to the relationship between arguments and predicates, roles can be defined as several types, such as core roles such as agents and recipients, as well as additional roles such as time, location, and goal.

Semantic role labeling is considered the foundation of many applications in natural language processing [2]. Unlike deep semantic information extraction, it does not process the entire text, but rather annotates roles on a sentence-by-sentence basis, and the annotated results can serve as the shallow meaning expressed in this sentence. Therefore, it has many advantages such as clear problem definition, strong operability, and easy utilization of analysis results. In addition, as an important component of event extraction, it can be well used in the construction of event graphs, and the quality of semantic role annotation results often determines the quality of the entire graph.

As a core technology of semantic analysis, semantic role annotation can perform specific processing on unstructured text, extract corresponding structured knowledge, and help build large-scale event graphs [3]. In addition, semantic role annotation, as the foundation of multiple natural language processing applications, if it can overcome current shortcomings and solve existing problems, it will provide beneficial assistance for these applications. This article utilizes the advantages of deep learning models and, based on existing work, optimizes and improves model algorithms. Based on the current task and language characteristics, a reasonable network structure is designed to automatically extract rich semantic features, achieving an improvement in model generalization ability while reducing resource utilization.

1.1. **Related work.** Due to the lack of available corpora, early semantic role annotation methods mainly relied on manually written rules and semantic dictionaries [4]. However, this approach required specific experts to build a large-scale knowledge base. As the amount of text data increased, language expression methods emerged endlessly, and limited semantic rules began to fail to meet the needs of problem-solving. Therefore, more rules needed to be constructed. However, contradictions and conflicts between different rules also emerged [5]. In recent years, with the release of some corpora, researchers have begun to attempt to use machine learning to handle semantic role annotation tasks.

The earliest method used to handle this task was through generative modeling [6], which believed that a series of features we see, such as part of speech, voice, etc., are generated by hidden features (semantic roles). Naive Bayes is a typical representative of this model. Its biggest advantage is its fast-processing speed and low dependence on training corpus. Its disadvantage is that it assumes that the various features of the sample are independent of each other (in reality, the features of the sample often have a certain degree of dependence), and it has high requirements for the expression form of input data, which is the main reason why it cannot achieve good performance in this task [7].

Discriminant models are a type of predictive model that directly learns the decision function $f(w)$ or conditional probability distribution $P(r|w)$ from data [8]. Typical representatives include linear interpolation, maximum entropy model, decision tree model, support vector machine, etc. Gildea and Jurafsky [9] first applied linear interpolation to the processing of semantic role annotation tasks based on existing work. Although this method did not achieve good results, the seven features proposed by Gildea were used as reference standards for classification by subsequent researchers.

The Maximum Entropy Model, as one of the most popular models in semantic role annotation tasks, is a classic classification algorithm. Its basic idea is to establish a model for all known facts without considering all unknown facts [10], which need identify all models that meet the conditional constraints to find the $P(r|w)$ corresponding to the maximum conditional entropy. Due to the convenience of setting some constraints, and the ability to enhance the model's adaptability to prediction data and fit training data through these constraints, it has obvious advantages in handling multi classification tasks. However, the number of constraint conditions is usually positively correlated with the number of training samples. Therefore, when the data volume is large, the computational complexity of the model will significantly increase, and the requirements for the experimental environment also need to be improved [11, 12].

The Decision Tree Model is a simple and easy-to-use non parametric classifier that selects a feature that constitutes a sample each time and classifies it based on the value of the sample. The selection of features mainly relies on three algorithms. The first two rely on the entropy model of information theory, while the third uses the Gini coefficient [13]. The advantage of the model is that it usually does not require pre-processing of samples

and can tolerate missing values in the data; When the model is too complex, pruning operations can be used to enhance the model's adaptability.

Overall, the two most important parts of machine learning based methods are feature acquisition and model selection. For feature acquisition, common methods include finding new features, combining existing features, and so on. However, this approach often leads to a series of problems, such as when the extracted feature has little to do with semantic role annotation, the addition of that feature not only does not promote the improvement of the results, but also increases the burden on the system, resulting in redundant calculations. In addition, when there are inherent issues with the feature labels used, such as incorrect part of speech labeling or incorrect syntax dependency labeling between various components in a sentence, these can further lead to the accumulation of errors and reduce the performance indicators of classification results. In addition, machine learning algorithms often fail to learn potential features between a large number of samples, which often play an important role in the classification of types. Therefore, the effectiveness of the model cannot continue to improve after reaching a certain level. However, due to the strong interpretability of machine learning, the trained models are often relatively stable and can ensure a certain level of accuracy. Therefore, it is still widely used in industry.

Deep learning, as a part of machine learning, was first proposed by G.E. Hinton in 2006 and defined as deep neural networks [14]. Unlike machine learning, deep learning is dedicated to building neural networks with specific structures to learn the intrinsic information patterns of samples from a large amount of training data. Therefore, it can not only help people free themselves from heavy feature engineering, but also effectively reduce the impact of feature recognition and construction on the system.

Classify the models involved in deep learning based on their structure, mainly including convolutional neural networks, recurrent neural networks, autoencoders, deep belief networks, etc. Due to the fact that we usually view sentences in text as sequences, the commonly used models in natural language processing are convolutional neural networks, recurrent neural networks, and their variants. Collabert et al. [15] first proposed a system framework based on convolutional neural networks in 2011 and applied it to four tasks: part of speech tagging, word segmentation, named entity recognition, and semantic role labeling, achieving good results. This system also became the foundation for using neural networks for natural language processing and annotation tasks in the future. Lin et al. [16] proposed an Attention Segmented Recurrent Neural Network (ASRNN) that relies on a layered attention neural Semi-Markov Conditional Random Field (semi CRF) model to perform sequence labeling tasks. Munir et al. [17] used integrated filter generation networks to process semantic information, widening the gap between syntax aware and syntax agnostic SRL systems. Cai and Lapata [18] proposed an LSTM based semantic role annotator that improves algorithm performance by jointly training with two auxiliary tasks.

The above studies are all aimed at the task of semantic role annotation in English. Due to the complexity of Chinese itself and the insufficient training data, the progress of the work is relatively slow. Most research still focuses on machine learning methods. At present, deep learning-based methods mainly rely on bidirectional LSTM and incorporate some other features as inputs to improve the model's prediction results. Wan et al. [19] proposed a model based on Bidirectional Long Short-Term Memory Network and Conditional Random Field (Bi-LSTM-CRF). Yang and Zong [20] embedded lexical and grammatical information into the feature vectors of each independent variable, and then used K-means to cluster all feature vectors in the training set. This method proved to significantly improve the performance of SRL.

1.2. **Motivation and contribution.** Overall, semantic role annotation methods for Chinese and Russian based on deep learning still cannot be fully end-to-end, and certain feature information still needs to be combined during processing. And there is still a certain gap between the results obtained and English. Therefore, in response to the current shortcomings, we believe that the current method to improve semantic role annotation is to improve the existing model structure, so that it can fully utilize existing resources and maximize the mining of linguistic features contained in the text.

This article mainly studies how to improve the performance of the task of Chinese semantic role annotation. Through extensive field research, analyze the legacy issues of existing models and make improvements based on them. For semantic role annotation in Chinese, we have mainly made the following improvements: (1) Use the Highway Bidirectional Long Short-Term Memory (Highway Bi-LSTM) algorithm to replace the original Bi-LSTM, and combine it with the Self Attention mechanism to form the core layer of the model, in order to extract rich semantic information and grammar rules from the current dataset. (2) Using deep models to replace single-layer models further enhances the neural network's ability to mine text information. (3) The algorithm using Conditional Random Field (CRF) as the decoding layer can fully utilize sentence information to obtain the optimal label sequence.

In addition, we also explored the impact of model parameter selection on system performance in experiments, including the number of model core layers, word vector selection, decoding layer algorithms, etc. Through a series of comparative experiments, find a set of configurations that are more suitable for the current model.

2. Analysis of relevant principles.

2.1. Semantic role annotation theory.

2.1.1. *The definition of semantic role annotation.* Semantic role annotation is one of the various information extraction techniques and is considered an important step in promoting computer understanding of human language. It takes the sentence as the unit and the predicate in the sentence as the core, analyzes the dependency relationships between other components and predicates in the sentence, and classifies these components. For example:

[小明(Xiaoming) ARG0] [昨天(Yesterday) ARGM-TMP] [晚上(Evening) ARGM-TMP]
[在家(At Home) ARGM-LOC] [看(Watching) V] [电视(TV) Arg1]

Among them, " 看" is the predicate, which serves as the trigger for the event. " 小明" and " 电视" respectively refer to the agent and receiver in the sentence. " 昨天晚上" and " 在家" refer to the time and location of the action, respectively. In addition, semantic role labeling does not care about situations where the predicate changes but the sentence meaning remains unchanged, such as:

[贝多芬创作了摇篮曲(Beethoven created the lullaby)]
[摇篮曲的作者是贝多芬(The author of the lullaby is Beethoven)]

Although the semantics expressed by the two are the same, the predicates are different, so the annotation results are also different. Due to the fact that predicates in Chinese cannot reflect tense information, annotation results may result in some information loss, such as:

[小明要去美国了(Xiaoming is going to the United States)]
[小明去美国了(Xiaoming has gone to the United States)]

These two sentences express different meanings, but the result of role labeling is the same. Therefore, in practical applications, it is often necessary to further process the results according to the situation. Due to the lack of in-depth research on the meaning expressed by sentences in semantic role annotation tasks, compared with event extraction, it has the advantages of clear problem definition, strong operability, and easy utilization of analysis results. It plays an important role in many applications in the field of natural language processing, such as question answering systems [21], machine translation [22], etc.

2.1.2. *Semantic role annotation corpus and representation.* This article uses the Chinese Proposal Bank (CPB) corpus as experimental data, and its labeling method refers to the English PropBank corpus. Figure 1 shows an example of labeling a sentence in CPB. Semantic roles are mainly divided into three categories, namely predicates, core arguments, and adjuncts arguments. Among them, the predicate is the triggering part of the event, and other roles are determined based on the predicate. The core argument refers to the parts that are directly related to the predicate, with a total of 6 types, specifically represented as ArgN, $N \in 0, 1, 2, 3, 4, 5$. ARG0 represents the agent, ARG1 represents the receiver, and ARG2-5 may have different meanings depending on the predicate. The additional argument refers to the parts that are not directly related to the predicate, with a total of 12 types, usually represented by ARGM-XXX, where XXX represents its specific meaning.

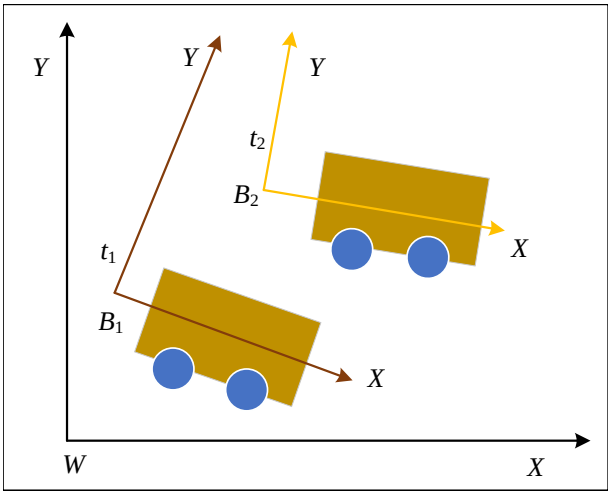

Figure 1. CPB sentence annotation example

In addition, in order to better identify the boundary information of semantic roles, some standard boundary labels are usually used, such as IOB representation, IOBES representation, etc. In the IOB notation, the first word of a phrase block starts with B, the remaining words start with I, and O represents the non semantic role outside the phrase block. The IOBES notation adds two additional labels on top of the IOB notation. When a phrase block contains only one word, the word starts with S. When the number of words in the phrase block is greater than 1, the last word starts with E.

2.2. **Dropout principle and application.** Due to the increasing complexity of the problem to be solved, simple neural networks cannot learn knowledge well from a large amount of data, and models often experience underfitting. Therefore, building more complex large-scale deep neural networks to solve problems has become a trend, usually by increasing the number and dimension of hidden layers. The deep model has multiple

nonlinear hidden layers, which can effectively learn the complex mapping relationship between input and output. However, due to limited training data provided, the features of noise in the samples will also be learned, and this part of the noise is not present in the test set. Therefore, overfitting occurs.

Dropout technique is a technique proposed by Hinton et al. in 2012 to prevent overfitting. Due to its advantages of clear definition, simple operation, and low computational cost, it is currently widely used in various deep neural network models. We know that combining multiple networks is a good way to prevent overfitting, but this approach of constructing different network structures or training the same network through different training sets to obtain different models often encounters some obstacles. Firstly, finding suitable parameters for these models through multiple trainings requires a significant amount of computational resources; Secondly, in reality, there is often a lack of sufficient datasets to train different networks.

Dropout technology can effectively solve the above two problems. During each training process, randomly select some hidden units and temporarily stop their work. The specific method is to disconnect their corresponding input and output connections. Each unit is independent of each other and is retained with a certain probability. Therefore, if a network has n units, we can consider it as a set of 2^n sub networks.

Due to the different hidden units randomly selected each time, it means that the units in the entire network are unrelated to each other. Every neuron must learn some useful information and cannot rely solely on other neurons, as it cannot guarantee that it will work simultaneously with a fixed neuron. This approach will make each neuron more robust, without relying on other neurons to correct its errors in learning.

Unlike training, during testing, some units are not randomly selected for discarding. The actual approach is to retain all hidden units and change their corresponding weights. For example, we obtain weights $w_{i,j}$ between two hidden units through training, indicating that hidden unit i points to hidden unit j. If the hidden unit i is retained with probability p, then during testing, the weight value between the two becomes pw_{ij}. This approach ensures consistency between test output and training output, and compared to other methods of reducing complexity, it can effectively reduce generalization errors.

The network calculation formula using Dropout is shown as follows:

$$r^{(l)} \sim \text{Bernoulli}(p) \tag{1}$$

$$\tilde{y}^{(l)} = r^{(l)} * y^{(l)} \tag{2}$$

$$z^{(l+1)} = W^{(l+1)}\tilde{y}^{(l)} + b^{(l+1)} \tag{3}$$

$$y^{(l+1)} = f(z^{(l+1)}) \tag{4}$$

Before using the Dropout mechanism, the neural network only needs to perform the two steps corresponding to Equation (3) and (4). After using Dropout, we need to first build several sub networks in some way. The specific method corresponds to Equation (1) and (2). In Equation (1), $r_j^{(l)}$ represents the value corresponding to the j-th hidden unit in the l-th layer, which can be randomly generated through the Bernoulli function. The probability of a value of 1 is p, and the probability of 0 is $(1-p)$. Then, in the same way, the values corresponding to other units in the same layer are obtained, and the corresponding results are represented in vector form, i.e. $r^{(l)}$. $y^{(l)}$ represents the units in the l-th layer. We multiply $r^{(l)}$ and $y^{(l)}$ so that some hidden units in the l-th layer have a value of 0, temporarily stopping their operation. Afterwards, apply the newly created neural network for training.

In our model, we set a Dropout layer after each Highway Bi-LSTM layer and retain hidden units with a probability of 0.5. We also set it before the CRF layer, with a value

of p set to 0.8. This is because we consider that the Encoder result contains a lot of information, so we cannot use a higher probability to prevent excessive information loss.

2.3. **BatchNorm method and its application.** When evaluating test data using trained deep learning models, we always assume that the training data and test data have the same distribution, which is called the independent and identically distributed assumption. The function of BatchNorm is to ensure that the inputs at each layer of the model have the same distribution.

Because the current model structure has undergone significant changes compared to before. The number of layers in the model increases, and during the training process, the input of each layer undergoes a certain transition (the overall distribution of data approaches the upper and lower limits of the activation function). When using backpropagation algorithm for calculation, the obtained gradient is very small, resulting in the network parameters not being updated and the training speed slowing down. To avoid this issue, we use BatchNorm technique to transform the distribution of each layer back to a normal distribution with a mean of 0 and a variance of 1. But this approach will result in the training effect of deep networks being consistent with that of single-layer networks, because implementing BatchNorm for each layer is equivalent to performing multiple linear transformation operations. The results of multiple linear transformation operations can be represented by a single linear transformation, which means that the model's expressive power is greatly limited.

In order to enhance the expressive power of the neural network, this method also uses two parameters, scale and shift, to make some adjustments to the results. Below, we will introduce the specific approach of BatchNorm. Assuming a deep neural network has two hidden layers, we need to perform BatchNorm on the activation values of each hidden layer unit. We can assume that we first perform a normalization operation on the input of each layer to make it a distribution with a mean of 0 and a variance of 1. The specific operations are shown as follows:

$$\hat{x}_t^{(k)} = \frac{x_t^{(k)} - \mathbb{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}} \tag{5}$$

The $x_t^{(k)}$ here does not refer to the output of the $t-1$ layer network, but rather to a linear transformation of it. $\mathbb{E}[x^{(k)}]$ refers to the mean of all data in one training session, and $\mathrm{Var}[x^{(k)}]$ refers to the variance of all data in one training session.

After the above transformation, the activation values of each hidden layer neuron follow a normal distribution, which can increase the corresponding derivative value when using backpropagation for training, avoiding the problem of gradient vanishing. However, as we mentioned earlier, this approach weakens the role of the activation function and greatly reduces the expressive power of the model. Therefore, we still need to use two additional parameters to make some adjustments to the results. The specific implementation is shown as follows:

$$y^k = \gamma^k \hat{x}^{(k)} + \beta^k \tag{6}$$

where γ^k and β^k can be trained to obtain optimal values.

3. A Chinese semantic role annotation model based on Highway Bi-LSTM.

3.1. **Model framework.** In this chapter, we will introduce the Chinese semantic role annotation model and its corresponding components, as shown in Figure 2. We can consider it as a sequence annotation task, where the input of the model includes a sentence that has already been divided into words and its corresponding predicate. To mark the

predicate, the usual practice is to set the corresponding part of the predicate to 1 and the other words in the sentence to 0. In our model, in order to enrich the representation of predicates, we set a window centered on the predicate with a size of $(s \times 2 + 1)$, where s represents the number of words located on the left/right sides of the predicate. We set the words inside the window to 1 and the rest to 0. The output of the model is also a sequence, and each element in the sequence is labeled as a corresponding role.

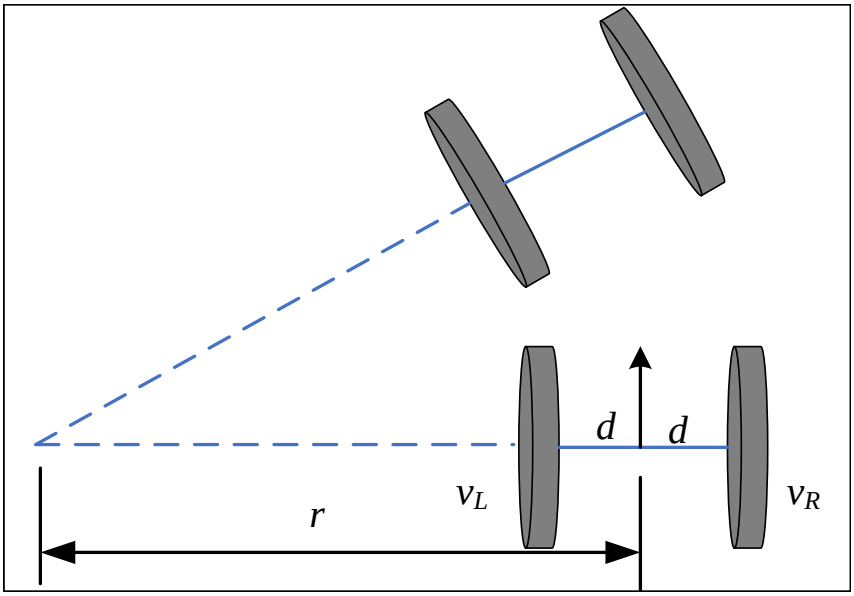

Figure 2. Chinese semantic role model framework

From the graph, we can see that the framework consists of multiple network layers with the same structure, which we refer to as the core layer. Each core layer also includes two sub layers, each using the Highway Bi-LSTM and Self Attention mechanisms. For the decoding layer, we choose to use the conditional random field algorithm to determine the optimal annotation result for the sentence.

The core layer structure is shown in the dashed box in Figure 2, which can be seen to be composed of two sub layers. Some of them use Highway Bi-LSTM technology, while the other part uses Self Attention technology. Sections 3.2 and 3.3 respectively provide detailed explanations of the principles and effects of these two technologies.

3.2. **Deep layers Highway Bi-LSTM.** Bidirectional LSTM is another variant of recurrent neural networks. In recent years, various literature has shown that it can effectively solve the gradient vanishing problem in recurrent neural networks. Many previous experiments have also shown that bidirectional LSTM can handle semantic role annotation tasks well. Therefore, in this article, we still use bidirectional LSTM as the core component of our model.

Compared to GRU, LSTM utilizes three gates to control information. They are Forget Gate, Input Gate, and Output Gate, respectively. The forgetting gate is better than the reset gate in GRU, which can selectively discard redundant information that the model considers. The input gate can filter new features input by the model, which is not present in the GRU model. The input gate and forget gate correspond to the weights of the current feature information and the previous state information, respectively. The output gate is used to determine the final output of the model. The corresponding formula is shown as follows:

$$i_{l,t} = \sigma(W_i^l[h_{l,t-1}, x_{l,t}] + b_i^l) \tag{7}$$
$$o_{l,t} = \sigma(W_o^l[h_{l,t-1}, x_{l,t}] + b_o^l) \tag{8}$$
$$f_{l,t} = \sigma(W_f^l[h_{l,t-1}, x_{l,t}] + b_f^l) \tag{9}$$
$$\tilde{c}_{l,t} = \tanh(W_c^l[h_{l,t-1}, x_{l,t}] + b_c^l) \tag{10}$$
$$c_{l,t} = i_{l,t} \circ \tilde{c}_{l,t} + f_{l,t} \circ c_{l,t-1} \tag{11}$$
$$h_{l,t} = o_{l,t} \circ \tanh(c_{l,t}) \tag{12}$$

Because one-way networks can only learn dependency information in a sentence from one direction, the components in the sentence are not only related to the previous content, but also closely related to the subsequent content. Therefore, this article applies bidirectional LSTM to learn features from two directions. The specific implementation is shown as follow:

$$h_t^{\rightarrow} = \text{LSTM}(x_t, h_{t-1}^{\rightarrow}) \tag{13}$$
$$h_t^{\leftarrow} = \text{LSTM}(x_t, h_{t+1}^{\leftarrow}) \tag{14}$$
$$h_t = h_t^{\rightarrow} \oplus h_t^{\leftarrow} \tag{15}$$

Semantic role annotation is a relatively complex task, and it is difficult to fully extract the feature information present in the training samples by relying solely on single-layer neural networks. Due to the ability of deep neural networks to capture more discriminative information than shallow neural networks, the performance of the model is often better. Therefore, this article decides to attempt to increase the number of network layers, that is, to increase the depth of the model from a spatial perspective, in order to improve the annotation ability of the model.

One problem directly caused by the deepening of network layers is the problem of vanishing or exploding gradients, which slows down the training speed of the network and results in the loss of previous information. Based on this, this article uses bidirectional LSTM technology based on Highway connection, and the specific principle is shown as follow:

$$\text{new}h_{l,t} = T \cdot h_{l,t} + (1 - T) \cdot x_{l,t} \tag{16}$$

where T represents the "conversion gate", $h_{l,t}$ represents the output of Bi-LSTM in the l-th layer at time t, and new$h_{l,t}$ represents the new output result after using the conversion gate.

3.3. **Self attention mechanism.** The concept of Self Attention originated from the paper "Attention Is All You Need", authored by Google's machine translation team. It has been proven to be a new method that can improve the performance of numerous natural language processing applications, with specific usage scenarios including machine translation, reading comprehension, text summarization, etc. We have previously proposed that bidirectional LSTM can effectively capture contextual information present in sentences, but it still has certain limitations when processing this task. LSTM processes sequences at a certain time step, and when processing subsequent information, it often needs to wait for the previous information to be processed; In addition, this approach often overlooks the tree structure information present in the sequence. In reality, the interior of most sentences always presents a tree like structure.

Self Attention, also known as Intra Attention, is an attention mechanism used to handle the internal elements of a sequence, which can directly characterize the global dependencies of the entire sentence. The biggest advantage of this mechanism is that it can obtain the dependency relationship between any two words in a sentence without considering distance, while learning the internal structural information in the sentence. It reflects the feature information in sentences in a more convenient and flexible way. In our model, we combine this mechanism with deep Highway Bi-LSTM to obtain a richer representation of information.

Assuming we want to apply the Self Attention mechanism to a sentence that has already been divided into words, we can first assume that each word in the sentence is composed of a series of $\langle \text{Key}, \text{Value} \rangle$ pairs. Now we need to process one of the words, calculate the correlation between all words in the sentence, and the result we obtain is called attention value. The first step in implementing this mechanism is to generate three corresponding vectors for each word, namely $Query_t$ (Q_t), Key_t (K_t), and $Value_t$ (V_t). The subscript t indicates that these are the three calculation results of the t-th word. This section is obtained by multiplying a vector H_t by three weight vectors ($W_{q,t}$, $W_{k,t}$, $W_{v,t}$), which are the parameters that the model needs to train. Here, H_t is the execution result of the t-th time step in Highway Bi-LSTM. The specific implementation is shown as follow:

$$Q_t = H \cdot W_{q,t} \tag{17}$$

$$K_t = H \cdot W_{k,t} \tag{18}$$

$$V_t = H \cdot W_{v,t} \tag{19}$$

The next step is to use the current word to be processed as the target vector for the Query, and calculate the similarity with the Key vectors of all words (including the word itself) in the sentence. Assuming that we need to calculate the attention value between the i-th word and other words, where Q_i represents the target vector for this word and K_j represents the Key vector for the j-th word, we need to calculate the dot product of Q_i and each K_j. The value range of j is $j \in \{1, 2, 3, \ldots, M\}$, where M is the length of the sentence. Then we scale and normalize the calculation results to obtain the corresponding attention values. Finally, we multiply these attention values with the corresponding Value vectors to obtain the weighted vector of the current word. In practical applications, in order to accelerate the calculation speed, matrix calculation is usually used, which combines each input vector into a matrix, and the corresponding weight vectors are also merged into a matrix form. Therefore, our entire description process can be expressed as follow:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^{\text{T}}}{\sqrt{d}}\right)V \tag{20}$$

where d represents the dimension of K or Q.

In order to enhance the model's ability to focus on different positions and enrich the representation of subspaces, we also adopted a multi head attention mechanism, which essentially uses multiple sets of Q, K, and V vectors for multiple calculations, similar to the fusion mechanism we mentioned earlier. We used 8 sets of such vectors for training. In the end, the calculation results of multiple Attention are merged into a single matrix, and then multiplied with a weight matrix to obtain a fixed dimensional result. The corresponding "Multi Head" attention mechanism can be represented as follow:

$$\text{head}_i = \text{Attention}(QW_q^i, KW_k^i, VW_v^i) \tag{21}$$

$$\text{Head}' = \text{Concat}(\text{head}_1, \ldots, \text{head}_n) \tag{22}$$

where head_i here represents the result of the i-th Attention, and Concat represents the connection operation.

3.4. **Conditional random field.** The above neural network module is called the Encoder part of the model, and its main function is to extract features from the input sequence data. The generated data structure is still a vector sequence. However, we hope that the model can output discrete label results, so we need to set up a decoder to decode the output part. In this section, we used conditional random fields as our final decoding layer algorithm.

When decoding, CRF can comprehensively consider sentence level label information and then use the Viterbi algorithm to obtain the optimal label sequence. CRF calculates a score for each possible label sequence starting from the entire sentence, and then normalizes the generation probability of each label sequence through the Softmax function. The implementation principle is shown as follows:

$$s(X, y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i} \tag{23}$$

$$P(y|X) = \frac{e^{s(X,y)}}{\sum_{\tilde{y} \in Y_x} e^{s(X,\tilde{y})}} \tag{24}$$

where $s(X, y)$ corresponds to the score when the output sequence is y, $A \in \mathbb{R}^{k \times k}$ represents the role transition matrix here, and k is the total number of role types. $A_{i,j}$ represents the loss of transferring from role i to role j. $P \in \mathbb{R}^{n \times k}$ is the probability of a character being generated, and $P_{i,j}$ represents the probability that the i-th word in the sentence is defined as the j-th character. In Equation (24), Y_x represents all possible character sequences corresponding to the input sentence X. $P(y|X)$ represents the conditional probability that when the input is X, the output is y, and y is the actual character sequence corresponding to X. The goal of this algorithm is to maximize $P(y|X)$. Therefore, the corresponding loss function is shown in Equation (25). For model training, we can use backpropagation algorithm to maximize the log function of character generation probability:

$$\log P(y|X) = s(X, y) - \log \left(\sum_{\tilde{y} \in Y_x} e^{s(X,\tilde{y})} \right) \tag{25}$$

4. Experiment.

4.1. **Data set.** The dataset used in this article is part of the corpus CPB constructed by the University of Pennsylvania. CPB is a shallow semantic annotation resource based on Chinese Treebank (CTB) annotation by Upenn, which adds semantic information expressed by the predicate argument relationship in CTB. The annotated data of CPB mainly comes from Xinhua News Agency, Sinorama News Magazine, blogs, radio interviews, etc. This corpus contains a total of 18 semantic roles (6 core roles + 12 additional roles). Due to the use of IOBES notation in our experiment, the number of labels increased to 66. The dataset used in our experiment contains a total of 20972 sentences, including 17839 sentences in the training set, 1115 sentences in the validation set, and 2018 sentences in the test set. Considering that the sentences in the test set were not annotated, we have decided to use only the validation set to evaluate the effectiveness of our model.

Table 1. Experimental environment description

Hardware environment	Software environment
CPU: Intel(R) Xeon(R) Gold 6126 @2.60GHz	OS: Ubuntu 16.04
Memory: 192GB	Python: 3.5
GPU: Tesla V100, 32GB	Tensorflow-gpu: 1.6.0

4.2. **Experimental configuration.** The experimental environment configuration is shown in Table 1.

For the parameter setting part of the model, the weights corresponding to neurons in all sub layers were initialized using the Xavier initializer, and for other parameters in the model, we used a normal distribution function to set the initial values. We used pre-trained word vectors with dimensions of 300 and People's Daily as the corpus. A large amount of literature has proven that using word vectors can effectively accelerate model training speed and execution effectiveness. We will discuss the impact of word vectors on this task later. Through multiple experimental comparisons, we have obtained a relatively good set of model hyperparameter values. The specific values are shown in Table 2. We set the number of training rounds to 10000, and every 400 steps, we will test the validation set with the current model to evaluate its performance. In addition, we also used the Early Stopping strategy during training to prevent overfitting of the model.

Table 2. Experimental parameter configuration instructions

Model parameter	Value
Word vector dimension	300
Predicate marker dimension	200
Number of hidden units in Bi-LSTM	300
Number of core layers	3
Optimizer	Adam
Learning rate	0.001
Batchsize	128

4.3. **Evaluating indicator.** Because when dealing with this task, we do not treat it as a multi classification problem, but rather as a sequence labeling problem. Consistent with most deep learning-based methods, we also use F1 value as the evaluation metric for the model. The calculation result is represented by the following formula:

$$\text{precision} = \frac{A}{B} \tag{26}$$

$$\text{recall} = \frac{A}{C} \tag{27}$$

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{28}$$

where A represents the number of correctly labeled words in the entire sample, B represents the total number of semantic roles annotated by the model, and C represents the total number of semantic roles in the test data.

4.4. **Experimental results and analysis.** To demonstrate that the current model indeed has a certain improvement effect, we compared it with the CSRL-BRNN model. To ensure the effectiveness of the results, we reconstructed its model according to its description in the paper. Afterwards, we trained the two models using the same data in the same experimental environment. In the end, the result obtained by the CSRL-BRNN model was $F_1 = 75.6\%$. The experimental comparison in Figure 3 shows that our model has improved the F1 value by 5.9%. The reason may be that: (1) our model uses deep Highway Bi-LSTM instead of single-layer Bi-LSTM; (2) Our model enriches the representation of predicates by replacing them with all words within the central window of the predicate; (3) Self Attention helps the model extract more representative features; (4) Using CRF as the decoding layer algorithm. Next, we will start analyzing some factors that affect the experimental results, as well as the optimal values determined through the experiment.

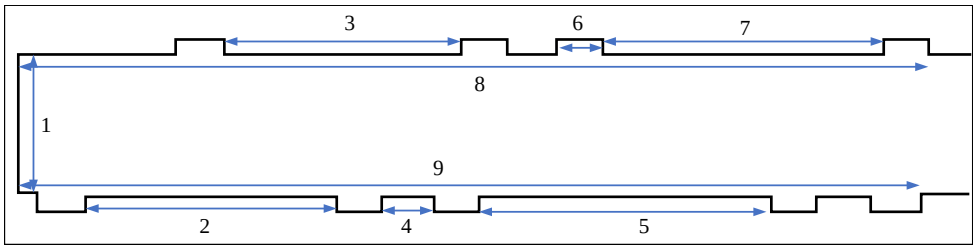

Figure 3. Comparison of experimental results

4.4.1. *The impact of word vectors on models.* In this study, we used different features for training and analyzed the relationship between the word vectors obtained and the model results. The experimental results are shown in Figure 4. Comparing random vectors, word feature vectors, and word feature vectors, it can be found that word feature vectors have the best performance, with an F1 value of the highest 81.2%.

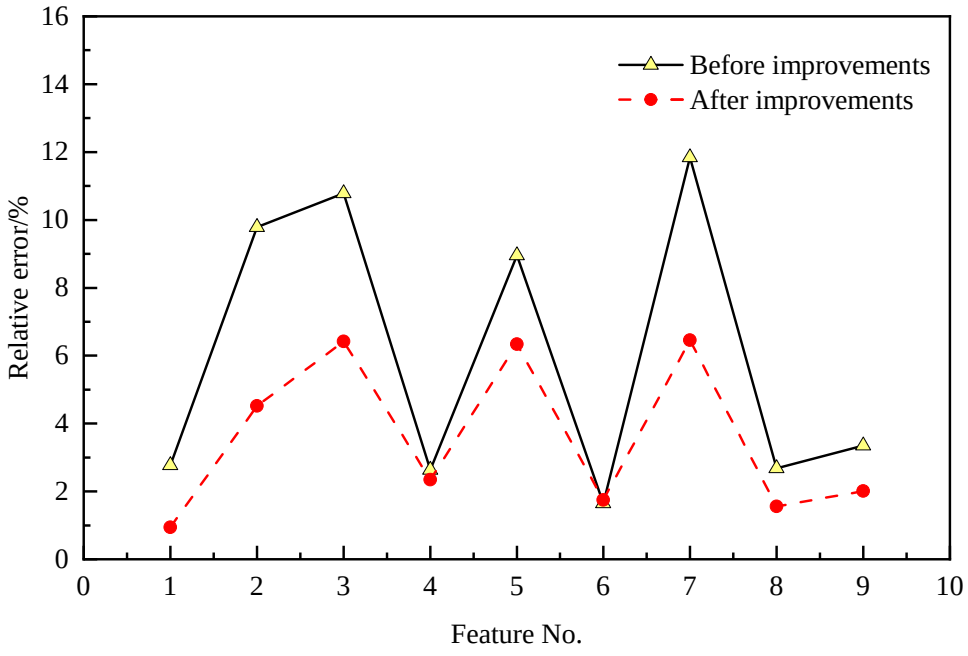

Figure 4. Comparison of results of different feature word vectors

4.4.2. *The impact of core layers on models.* Many previous literature has shown that increasing model depth can improve problem-solving ability, as increasing model depth can extract richer features. However, it is not that the higher the depth of the model, the better the model effect, and the best effect will be achieved when reaching a balance value. In this section, we will adjust the number and size of core layers, and find the most suitable parameters through multiple rounds of experimental results comparison. The experimental results are shown in Figure 5. In the end, we obtained the best model performance of 81.2% when the number of core layers is 3.

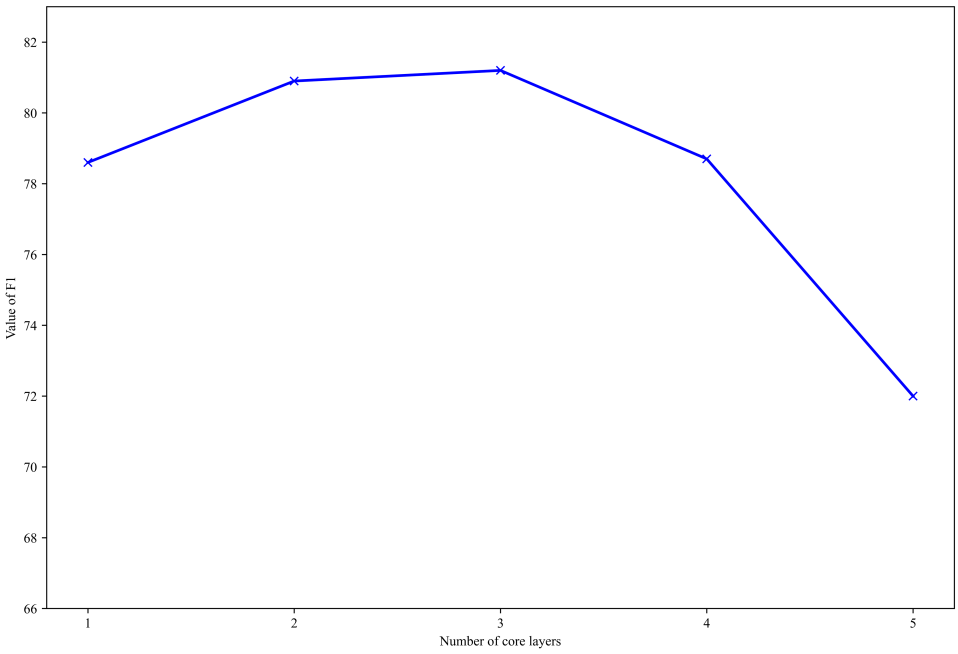

Figure 5. Comparison of results for different core layers

5. **Conclusions.** In this article, we constructed a Chinese semantic role annotation model based on Highway Bi-LSTM. Building a model for processing sequential data typically involves incorporating some feature engineering, which can lead to accumulated errors. Our approach is to maximize the ability of existing models to mine dataset information by improving them.

In this study, the model input was sentences that had already been segmented into words, and we did not consider the impact of segmentation on the model during the analysis process. Although the current segmentation technology has reached a high level, there are still certain errors that can lead to the accumulation of errors. Therefore, we will attempt to combine sentence segmentation with semantic role labeling to construct a completely end-to-end model.

REFERENCES

[1] M. Surdeanu, L. Màrquez, X. Carreras, and P. R. Comas, "Combination strategies for semantic role labeling," *Journal of Artificial Intelligence Research*, vol. 29, pp. 105-151, 2007.

[2] S. S. Pradhan, W. Ward, and J. H. Martin, "Towards robust semantic role labeling," *Computational Linguistics*, vol. 34, no. 2, pp. 289-310, 2008.

[3] K. Xu, H. Wu, L. Song, H. Zhang, L. Song, and D. Yu, "Conversational semantic role labeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2465-2475, 2021.

[4] Y. Ma, Y. Peng, and T.-Y. Wu, "Transfer learning model for false positive reduction in lymph node detection via sparse coding and deep learning," *Journal of Intelligent & Fuzzy Systems*, vol. 43, no. 2, pp. 2121-2133, 2022.
[5] L. Soulier and L. Tamine, "On the collaboration support in information retrieval," *ACM Computing Surveys*, vol. 50, no. 4, pp. 1-34, 2017.
[6] S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky, "Support vector learning for semantic argument classification," *Machine Learning*, vol. 60, pp. 11-39, 2005.
[7] K. Toutanova, A. Haghighi, and C. D. Manning, "A global joint model for semantic role labeling," *Computational Linguistics*, vol. 34, no. 2, pp. 161-191, 2008.
[8] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2761-2773, 2010.
[9] D. Gildea, and D. Jurafsky, "Automatic labeling of semantic roles," *Computational Linguistics*, vol. 28, no. 3, pp. 245-288, 2002.
[10] J.-D. YU, X.-J. WANG, and Z.-T. YU, "Semantic Role Labeling Based on Maximum Entropy Model," *Microelectronics & Computer*, vol. 27, no. 8, pp. 173-176,180, 2010.
[11] N. Xue, "Labeling Chinese predicates with semantic roles," *Computational Linguistics*, vol. 34, no. 2, pp. 225-255, 2008.
[12] Y. He, Y. Li, L. E. Meng, and H. Xu, "Towards patent text analysis based on semantic role labelling," *International Journal of Computational Science and Engineering*, vol. 15, no. 3-4, pp. 256-266, 2017.
[13] A. Moschitti, D. Pighin, and R. Basili, "Tree kernels for semantic role labeling," *Computational Linguistics*, vol. 34, no. 2, pp. 193-224, 2008.
[14] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527-1554, 2006.
[15] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493−2537, 2011.
[16] J. C.-W. Lin, Y. Shao, Y. Djenouri, and U. Yun, "ASRNN: A recurrent neural network with an attention model for sequence labeling," *Knowledge-Based Systems*, vol. 212, 106548, 2021.
[17] K. Munir, H. Zhao, and Z. Li, "Adaptive convolution for semantic role labeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 782-791, 2021.
[18] R. Cai, and M. Lapata, "Syntax-aware semantic role labeling without parsing," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 343-356, 2019.
[19] F. Wan, Y. Yang, D. Zhu, H. Yu, A. Zhu, G. Che, and N. Ma, "Semantic role labeling integrated with multilevel linguistic cues and Bi-LSTM-CRF," *Mathematical Problems in Engineering*, vol. 2022, pp. 1-8, 2022.
[20] H. Yang, and C. Zong, "Learning generalized features for semantic role labeling," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 15, no. 4, pp. 1-16, 2016.
[21] S. J. Athenikos, and H. Han, "Biomedical question answering: A survey," *Computer Methods and Programs in Biomedicine*, vol. 99, no. 1, pp. 1-24, 2010.
[22] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493−2537, 2011.