Hindawi
Advances in Multimedia
Volume 2022, Article ID 5439320, 8 pages
https://doi.org/10.1155/2022/5439320

Research Article

Cross-Linguistic Similarity Evaluation Techniques Based on Deep Learning

Jun Li, Jing Zhang, and Mengjie Qian

Hebei Vocational University of Technology and Engineering, Xingtai 054035, China

Correspondence should be addressed to Jun Li; xpclijun@163.com

Received 3 March 2022; Revised 11 April 2022; Accepted 18 April 2022; Published 16 May 2022

Academic Editor: Qiangyi Li

For the cross-linguistic similarity problem, a twin network model with ordered neuron long- and short-term memory neural network as a subnet is proposed. The model fuses bilingual word embeddings and encodes the representation of input sequences by ordered neuron long- and short-term memory neural networks. Based on this, the distributed semantic vector representation of the sentences is jointly constructed by using the global modelling capability of the fully connected network for higher-order semantic extraction. The final output part is the similarity of the bilingual sentences and is optimized by optimizing the parameters of each layer in the framework. Multiple experiments on the dataset show that the model achieves 81.05% accuracy, which effectively improves the accuracy of text similarity and converges faster and improves the semantic text analysis to some extent.

1. Introduction

The role of text similarity assessment in sentiment analysis, content recommendation, data mining, protection of intellectual property rights of electronic texts, and combating illegal copying and plagiarism of academic results cannot be underestimated. Since the launch of China Knowledge Network's "Academic Misconduct Literature Detection System" [1], it has blocked the publication of articles with a high repetition rate at the source of academic paper results; CrossCheck [2], an antiplagiarism literature detection system developed by the International Publishing Links Association, has minimized the textual repetition rate of English publications. With the increased degree of cultural exchange in the world and the explosive growth of various language resources on the Internet, cross-lingual scenarios are becoming more common, leading to the increasingly urgent need for cross-lingual applications. In order to address the resulting technical barriers and achieve resource sharing across languages, academia and industry have been actively exploring cross-language natural language processing techniques [3]. Among them, cross-lingual text similarity evaluation is an important research element.

At present, the main cross-lingual text similarity algorithms are machine translation and statistical translation modelling approaches [4, 5], lexicon-based approaches [6–8], and parallel corpus-based approaches [9, 10]. The advantage of the statistical translation model-based approach for full-text machine translation is that it can be operated directly with off-the-shelf tools, and the disadvantage is that the goodness of the translation tool or model has a significant impact on the final result. The advantage of the lexicon-based approach is that it is simple and easy to operate, and the disadvantage is that it is limited by the lexicon's coverage of words, and the effect is not apparent. The advantage of the parallel corpus-based method is that it is reliable and accurate, but the disadvantage is that the calculation is complicated, slow, and not robust. The neural network-based computation method is the mainstream cross-linguistic similarity computation method at present. The main idea is to train a neural network to map cross-lingual word embeddings [11] to the same semantic space and obtain the similarity interest between texts by calculating the distance between vectors.

Barrón et al. [12] obtained a similarity measure by calculating the proportion of n-grams overlapping in two sentences. N-grams represent consecutive semantic units of length n in a given sequence, either words or characters. Gabrilovich [13] used explicit semantic analysis of the same concept in Wikipedia to obtain a vector representation of

words and then compared them. Ferrero et al. [14] used cross-lingual word vectors to compute cosine similarity to obtain similarity. Kusner et al. [15] proposed characterizing all the words of two texts as vectors. The similarity between the two texts is solved by moving all the words of one text to the minimum moving distance of the other text in the vector space; Guo et al. [16] used a bilingual dual encoder model to generate sentence embeddings, encoding both the source and target texts as sentence embeddings. An approximate nearest neighbour (ANN) search is first performed for each source text sentence to obtain multiple target sentences for each source sentence. Then, the text matching score is finally obtained by using the approximate nearest neighbor ranking of the source and target sentence matching, the absolute difference of the sentence position index, and the matching rank weighted with the normalized confidence score.

Most of the existing studies only consider a single text feature or fuse models for semantic features. To determine text similarity, we should consider multiple levels, such as word semantic information, contextual semantic information, and document information. Deep learning is usually based on a deep nonlinear network structure, which extracts information at different levels by processing the input data in layers and finally obtains the essential representation of the data. Word vector representation, sentence vector representation, and document vector representation can be obtained using deep learning techniques combined with different feature engineering methods. Therefore, this paper proposes integrating the word information and contextual semantic information into the sentence itself as its semantic representation. By combining multiple neural networks, the preprocessed sentence is used as the input of the frame to calculate the semantic similarity of sentences. On this basis, paragraphs are regarded as long sentences and used as computing units to evaluate cross-language similarity.

2. Introduction to Related Theories

2.1. Word Vectors. Early work considered words as individual atomic symbols and used one-hot representation to represent words as high-dimensional sparse vectors, but this suffers from "dimensional catastrophe" and "lexical divide". In order to represent the similarities and differences between words, researchers have proposed distributed representations of words, also known as word vectors, which represent words as low-dimensional dense vectors and thus measure the semantic relationships between words with the help of the distance between vectors.

Early work used singular value decomposition (SVD) to obtain word distribution representations. The basic idea is that if two words frequently appear in the same document, their semantic relationships will be similar to each other. Specifically, we first accumulate the frequency of word cooccurrence in documents from the corpus, construct the cooccurrence matrix (M), and then decompose M into singular value decomposition, where each row of U is the vector representation of the corresponding word. Although this method can obtain partial semantic or structural information, there are many problems, such as the dimensionality of the matrix being too huge and very sparse and the complexity of SVD being O(n4), resulting in a very time-consuming training time. Also, if new words are added, the dimensionality of the matrix will change frequently and requires retraining.

Currently, there has been much success with iteration-based training approaches, which do not directly perform statistics and then computation from massive datasets, but rather build iterative models to learn predictions. Specifically, word vectors are used as model parameters, and the model is trained by optimizing a specific objective. During the optimization process, errors are evaluated to update the parameters so that the learned parameters can be used for language models, which is a fundamental problem in natural language processing. The NNLM model is shown in Figure 1.

The whole model is divided into two main parts as follows.

(1) First is a linear embedding layer. It maps the input N-1 one-hot word vectors (distributed vectors) to N-1 distributed word vectors through a shared $D \times V$ matrix C, where V is the size of the dictionary and D is the dimension of the embedding vector (a priori parameters). VThe C matrix stores the word vector to be learned.

(2) The second part consists of a tan$h$ hidden layer and a softmax output layer to form a simple forward feedback network. This part maps the output $N-1$ word vector to a probability distribution vector of length V, estimating the conditional probability of words in the dictionary in the input context.

$$\begin{aligned} p(w_i|w_1, w_2, \ldots, w_{t-1}) &\approx \mathrm{f}(w_i, w_{t-1}, \ldots, w_{t-n+1}), \\ &= g(w_i, C(w_{t-n+1}), \ldots, C(w_{t-1})). \end{aligned} \tag{1}$$

We can adjust the model's parameters by minimizing the cross-entropy regularized loss function θ.

$$L(\theta) = \frac{1}{T} \sum t \ \log \ f(w_t, w_{t-1}, \ldots, w_{t-n+1}) + R(\theta), \tag{2}$$

(θ) is a regularization term training; the aim is to find the parameter θ that maximizes the log-likelihood function L.

2.2. Semantic Similarity. The connection between the concepts that computer language corresponds to in the real world is represented by things and the meanings these senses have, the interpretation and logical representation of something tangible in the data domain. For computer science, semantics is the user's interpretation of a computer representation (i.e., a symbol) used to describe the real world; that is, the way the user uses it to relate the symbolic representation of the computer to the real world.

There are two main methods to calculate the semantic similarity of words; one is to organize the concepts of related words in a tree structure to calculate; the other is mainly through the information of word contexts, using statistical modelling methods.

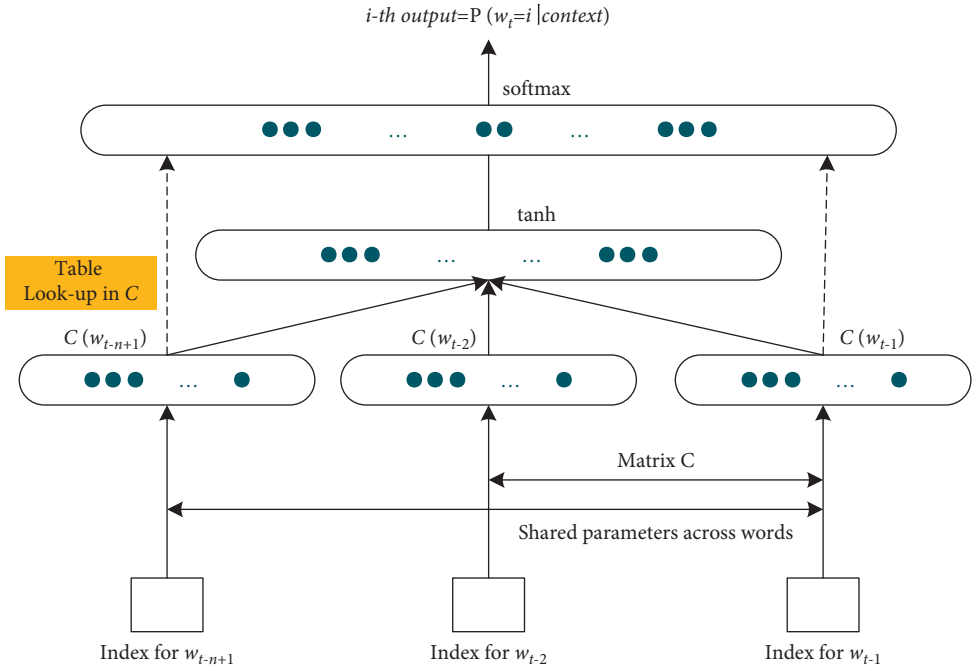

Figure 1: NNLM model.

2.3. Recurrent Neural Network. RNN is a class of recurrent neural networks that takes sequence data as input, recurses in the direction of sequence evolution, and all nodes are connected in a chain-like manner. RNN models can model text using contextual sequence relationships, thus solving the fixed window problem of feedforward neural network models (Figure 2). This is closer to the human brain's model of text processing than the feedforward neural network model, and using this correlation can improve the predictive power of the model.

However, the recurrent neural network itself also has a certain degree of limitation; the degree of learning for relatively distant sequence information will produce a longer distance dependence. The emergence of a long short-term memory neural network (LSTM) can better solve the dependence on the learning of relatively distant sequence information in recurrent neural networks. The LSTM structure is shown in Figure 3.

The implicit layer of the LSTM includes the memory cell, the input gate, the output gate, and the forgetting gate, and the cell and the three gates are connected. Let $ht-1$ and $ct-1$ be the output value of the implicit layer and the output value of the cell at the last moment, respectively, and it is the input value of the input layer at the current moment, and then, the update equations of the input gate, the forgetting gate, and the output gate of the LSTM are as follows:

$$\begin{aligned} i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right), \\ f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right), \\ o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o\right), \end{aligned} \tag{3}$$

where $\sigma(\bullet)$ is the function *sig*mod, W_{xi}, W_{hi}, and W_{ci} denote the connection weights between the input gate and

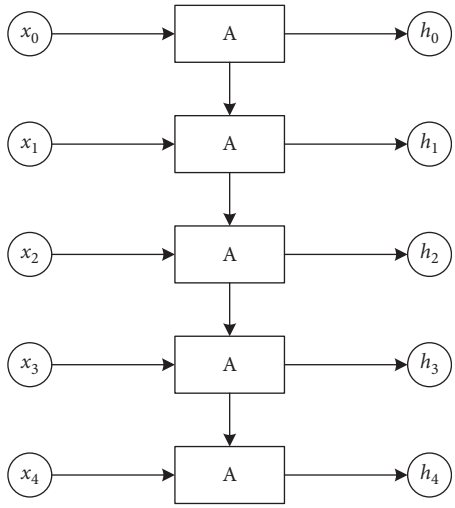

Figure 2: RNN structure diagram.

the input, implicit, and cell layers, respectively, and b_i denotes the bias value on the output gate; W_{xf}, W_{hf}, and W_{cf} denote the connection weights between the forgetting gate and the input, implicit, and output layers, respectively, and *fb* denotes the bias value on the forgetting gate; W_{xo}, W_{ho}, and W_{co} denote the connection weights between the input gate and the input, implicit, and output layers, respectively, and b_o denotes the bias value on the output gate. c_t denotes the candidate value of the cell at the current moment, and t_c is the actual state value of the cell at the current moment, calculated as follows:

$$\begin{aligned} \tilde{c}_t &= \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \end{aligned} \tag{4}$$

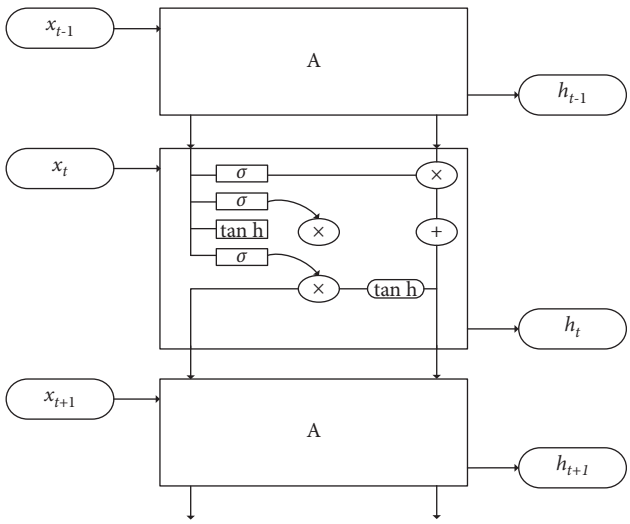

Figure 3: LSTM structure diagram.

where $\tanh(\bullet)$ denotes the hyperbolic tangent function, $\odot$ denotes the inner product operation. W_{xi} and W_{hi} denote the connection weights between the cell unit and the input layer and the hidden layer, respectively, and b_c denotes the bias value on the cell. Cell forgets the state value at the previous time according to the forgetting gate and updates the new input data according to the input gate. Finally, the output of the LSTM implicit layer at the current moment is

$$h_t = o_t \odot \tanh(c_t). \tag{5}$$

Both RNN and LSTM do not use the sequential information of neurons, but ON-LSTM neural network tries to use the sequential information of neurons in the neural network by forming an ordered structure of these disordered neurons with some changes to represent a specific information structure of sequence information. The ON-LSTM is similar to the LSTM model, except that the update function of the United States is excluded and replaced by a new update rule with the following expressions:

$$\begin{aligned}
\widetilde{\mathbf{f}}_t &= \overrightarrow{cs}\left(\text{softmax}\left(\mathbf{W}_{\widetilde{f}}\mathbf{x}_t + \mathbf{U}_{\widetilde{f}}\mathbf{h}_{t-1} + \mathbf{b}_{\widetilde{f}}\right)\right),\\
\widetilde{\mathbf{i}}_t &= \overleftarrow{cs}\ \left(\text{softmax}\left(\mathbf{W}_{\text{i}}\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_{\text{i}}\right)\right),\\
\boldsymbol{\omega}_t &= \widetilde{\mathbf{f}}_t \circ \widetilde{\mathbf{i}}_t,\\
c_t &= \omega_t \circ \left(f_t \circ c_{t-1} + i_t \circ \widehat{c}_t\right) + \left(\widetilde{f}_t - \omega_t\right) \cdot c_{t-1} + \left(\widetilde{i}_t - \omega_t\right) \cdot \widehat{c}_t,\\
\mathbf{h}_t &= \mathbf{o}_t \circ\ \tanh\left(\mathbf{c}_t\right).
\end{aligned} \tag{6}$$

3. Deep Learning Module

For the previous models considering only a single text feature or only for semantic features, this paper draws on the idea of literature [17] to form a twin network with On-LSTM as a subnet. Shared parameter configuration and connection weights are used to save network training time and also to avoid overfitting. The problem of long-distance dependence arises due to how the recurrent neural network learns sequence information at relatively long distances with the continuous input of relevant sequence information. The problem of perceptual weakening may arise for some information on the previous longer sequence information, which in turn can lead to gradient explosion and gradient dispersion.

In this paper, On-LSTM integrates the hierarchical structure (tree structure) of neurons into the LSTM after specific ordering, thus allowing the long short-term memory neural network to learn the hierarchical structure information in text sentences. In turn, the deep feature representation of the textual information and the semantic relationships encoded between each sentence is obtained, and On-LSTM allows the recurrent neural network model to perform tree synthesis without destroying its sequence representation. Subsequently, a fully connected network is used for higher-order semantic extraction, and the output is computed as the similarity result using a power finger function. The specific structure is shown in Figure 4.

The role of the On-LSTM representation layer is mainly to encode the input vectors from the preprocessing layer and map Chinese and English to the same semantic space as the input features of the higher-order semantic extraction layer. In this paper, we mainly use two-way On-LSTM for semantic encoding and combine the attention mechanism and cosine similarity for improvement. On-LSTM is the subnet structure for semantic representation and similarity calculation of bilingual sentences. An on-LSTM neural network can utilize the neuronal order information in the neural network by forming an ordered structure to represent some specific information structure of sequence information by some changes of disordered neurons. It can automatically learn the hierarchical structure information of the sentence, extract deep feature information from the contextual input information, and better learn the connection between the contexts.

In this paper, the whole framework is divided into four layers: the preprocessing layer, the On-LSTM representation layer, the higher-order semantic extraction layer, and the output layer. A twin neural network (Siamese network) is used as a weight-sharing network framework to fuse bilingual word embeddings and encode the input sequence representation by ordered neuron LSTM. Based on this, the distributed semantic vector representation of the sentences is jointly constructed using the global modelling capability of the fully connected network for higher-order semantic extraction. The final output part is the similarity of bilingual sentences, and the loss function optimally selects the parameters of each layer in the framework.

3.1. Data Preprocessing Layer. The processing layer can be considered the input source for the entire framework. The data preprocessing layer maps the input sentences as a list of word vectors as input to the implicit layer of the presentation layer On-LSTM. Commonly used preprocessing steps include word expansion and division, word stemming, POS, and destaying words. Only word splitting is required for semantic sentence modelling, and no lexical annotation is required.

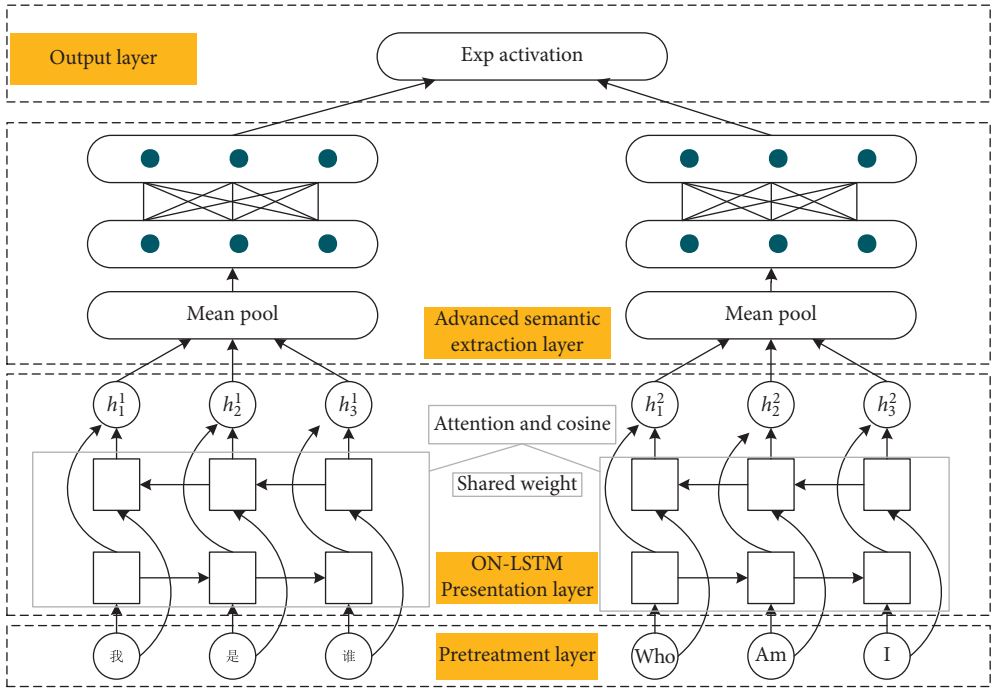

Figure 4: SCLSE structure diagram.

3.2. Semantic Encoding. The role of the On-LSTM representation layer is mainly to encode the input vectors from the preprocessing layer and map Chinese and English to the same semantic space as the input features of the higher-order semantic extraction layer. In this paper, we mainly use two-way On-LSTM for semantic encoding and combine the attention mechanism and cosine similarity for improvement. On-LSTM is the subnet structure for semantic representation and similarity calculation of bilingual sentences. An on-LSTM neural network can utilize the neuronal order information in the neural network by forming an ordered structure to represent some specific information structure of sequence information by some changes of disordered neurons. It can automatically learn the hierarchical structure information of the sentence, extract deep feature information from the contextual input information, and better learn the connection between the contexts.

3.3. Fusion Attention Mechanism and Cosine Similarity. The attention mechanism can make the model pay more attention to the representation of some words when expressing the semantic information of the text through specific computational methods. The attention mechanism assigns different weights ai to each word, and the magnitude of ai can reflect the importance of the word. This paper uses the context-aware attention mechanism proposed by Yang et al. [18]. This attention mechanism introduces a context vector, which helps identify entity words, and ai is randomly initialized and can be learned together with the weights of other attention layers.

3.4. Higher-Order Semantic Extraction. On top of the output of the On-LSTM representation layer, this paper chooses to use the mean pooling mechanism to fuse the final result information. The so-called mean pooling means that the output of each node in the On-LSTM representation layer is averaged with equal weights. Simply put, each word in the two sentences is voted on the final comparison result. The output of each On-LSTM can be interpreted as a judgment of whether the input words are semantically identical after observing the context. The results of this layer are noted as r_1 and r_2.

$$r_1 = \sum_{i=1}^{n} \frac{o_i}{n},$$
$$r_2 = \sum_{i=1}^{n} \frac{o_i}{n}. \tag{7}$$

3.5. Output of Similarity Results. The output of the fully connected layer is passed through the activation function to obtain the final result of the semantic similarity calculation of the sentences in both languages. The metrics to measure the difference of feature vectors are usually Euclidean distance, Manhattan distance, and cosine similarity. Here, in this paper, we adopt the formula used in the Manhattan LSTM model proposed by Mueller et al.

$$y = \exp\left(-c_1 - c_{21}\right). \tag{8}$$

The vector representation of the two sentences is obtained via the fully connected layer, and then, the difference is measured by the exponential function, which takes values in the range (0, 1] since the power order of the exponential function is the negative of the first-order parametrization. The biggest highlight of this formula is using the $L1$

parametrization to measure the difference of the obtained sentence vectors. Since the final output is the difference between the two-sentence vectors and can be viewed as a vector, the above function can be seen as the activation function for the final output.

4. Experiment and Analysis

The relevant hardware environment for the experiments is Windows 10 (64 bit), Intel (R) Core (TM) i7-6700HQ CPU @ 2.60 GHz processor, and 16.00 GB RAM. The development platform uses IntelliJ IDEA and PyCharm. The development languages are Python 3.6 and Java.

In order to verify the effectiveness of the algorithm in this paper, the accuracy, recall, and $F1$-score are selected as the evaluation indexes. The $F1$-score is the most common measure of the binary classification problem, the summed average of the precision and recall rates, with the maximum value being 1 and the minimum value being 0. Precision measures the model's accuracy, while recall measures the completeness of the model. Precision measures the accuracy of the model, while recall measures the completeness of the model, and the formula is as follows:

$$\begin{aligned} \text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}, \\ F1 &= 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \\ \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}. \end{aligned} \tag{9}$$

where TP indicates that the prediction is a positive sample, which is also a positive sample; FP means the prediction is a negative sample, but it is a positive sample; TN means the prediction is a negative sample; FN indicates that the prediction is a positive sample, but it is a negative sample. FN indicates that the prediction is a positive sample, but it is a negative sample.

By comparing the algorithm of this paper with CNN, LSTM, and BiLSTM-CNN, the results are shown in Table 1.

The framework proposed in this paper generally outperforms the other algorithms in computing text similarity. The accuracy and $F1$-score of the network model in this paper reach 81.05% and 75.26%, respectively, higher than the other four models. BiLSTM-CNN outperforms the single model due to the hybrid model. In this paper, the twin network composed of ON-LSTM automatically learns the hierarchical structure information better to represent the deep semantic information of the text, and the experimental results are also optimal. The results are shown in Figure 5.

The changing trend is to increase and then stabilize, with the CNN and LSTM models having the most similar rates of change compared to the other two groups of models. For the hybrid model of CNN and LSTM, BiLSTM-CNN outperforms any single model of both. The proposed text similarity

Table 1: Comparison results of the 4 models (%).

Model type	Accuracy	F1-score
CNN	77.10	66.47
LSTM	75.02	65.31
BiLSTM-CNN	79.36	74.45
Algorithm of this paper	81.05	75.26

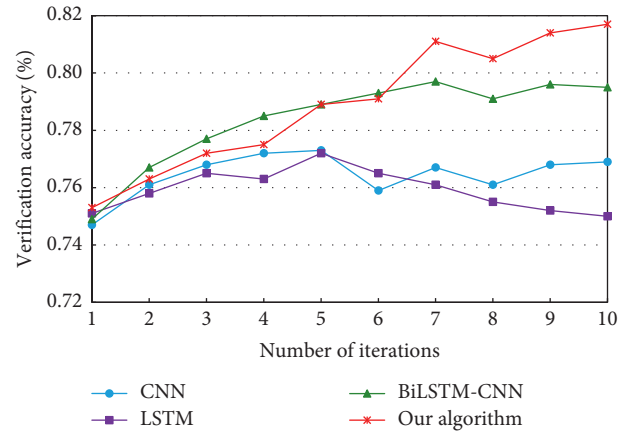

Figure 5: Trend of val_acc.

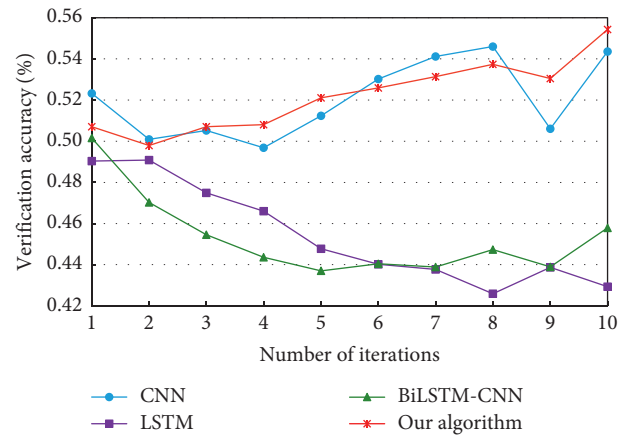

Figure 6: Trend of val_loss.

model shows an increasing and then smooth trend, which is better than the other four deep learning models and has better performance and higher accuracy in text feature extraction. The results are shown in Figure 6.

The fluctuation of the CNN model is the most unstable, and the overall fluctuation of this model is less compared with the other four groups of deep learning models. The trend of variation decreases and becomes stable. The short text similarity model based on ordered neuron LSTM has the advantages of fast convergence and high accuracy.

In order to measure the convergence speed among the five groups of deep learning models, the time cost of each iteration, that is, the time cost of the model to complete one training, is calculated. Figure 7 shows the trend of the time cost of different deep learning models with the number of iterations. Except for the CNN model, this model

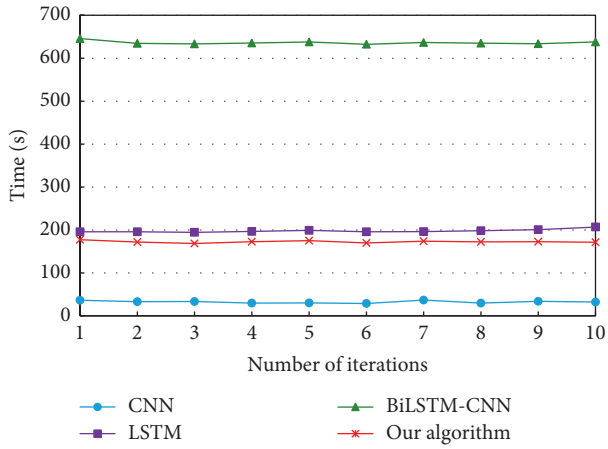

Figure 7: Time cost trend chart.

outperforms the other models in terms of time consumption, and the time cost is the smallest, which indicates that the short text similarity model based on ordered neuron LSTM proposed in this paper has the advantage of fast convergence speed.

5. Conclusion

In this paper, we propose a twin network structure model using On-LSTM as a subnet, mapping Chinese and English to the same semantic space, using ON-LSTM to automatically learn the hierarchical structure information better to represent the deep semantic information of the text, combining with the overall semantic information representation, and finally calculating the semantic similarity of the text through the matching layer. Several sets of comparison experiments show that the model in this paper has better improvement than other comparable models in index analysis and can better calculate the text similarity, which is feasible and effective.

Data Availability

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

Conflicts of Interest

The author declares that there are no conflicts of interest.

Acknowledgments

This work was supported by the Hebei Vocational University of Technology and Engineering.

References

[1] CNKI, "CNKI research center of scientific integrity management system. The intellectual property inspection system [EB/OL]," 2019, https://check.cnki.net/.

[2] Y. H. Zhang, "CrossCheck. An effective tool for detecting plagiarism," *Learned Publishing*, vol. 23, no. 1, pp. 9–14, 2010.

[3] A. K. Bhunia, P. P. Roy, A. Mohta, and U. Pal, "Cross-language framework for word recognition and spotting of Indic scripts," *Pattern Recognition*, vol. 79, pp. 12–31, 2018.

[4] G. Isao, U. Masao, S. Eiichiro, and S. Kurohashi, "Preordering using a target-language parser via cross-language syntactic projection for statistical machine translation," *Transactions on Asian and Low-Resource Language Information Processing*, ACM, vol. 14, no. 3, , pp. 13–35, 2015.

[5] H. Zhou, Y. Yang, Z. Liu, Y. Lin, P. Zhu, and D. Huang, "Jointly learning bilingual sentiment and semantic representations for cross-language sentiment classification," *Lecture Notes in Computer Science*, pp. 149–160, 2017.

[6] N. Ehsan, F. W. Tompa, and A. Shakery, "Using a dictionary and N-gram alignment to improve fine-grained cross-language plagiarism detection," in *Proceedings of the 2016 ACM Symposium on Document Engineering*, pp. 59–68P, ACM, New York, NY, USA, September 2016.

[7] Y. Niu, R. Xie, Z. Liu, and M. Sun, "Improved word representation learning with sememes," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 2049–2058, ACL, Vancouver, Canada, 2017.

[8] Y. Qu, T. Yang, and M. Wang, "Algorithm improvement of vocabulary semantic similarity with how net," in *Proceedings of the 8th ACM International Conference on Computer Modeling and Simulation*, pp. 114–118, ACM, Canberra, Australia, 2017.

[9] P. Rajendra, S. Sudeshna, and O. Philip, "Improving cross language information retrieval using corpus based query suggestion approach," in *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 448–457P, Springer, Cairo, Egypt, 2015.

[10] F. Sadat, A. Maeda, and M. Yoshikawa, "cross-language information retrieval via dictionary-based and statistical-based methods," in *Proceedings of the IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, pp. 595–598P, IEEE, Victoria, Canada, 2015.

[11] A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proceedings of the 13th Aaai Conference on Artificial Intelligence*, AAAI Press, Phoenix, AZ, USA, 2016.

[12] A. Barrón-cedeno, P. Rosso, E. Agirre, and G. Labaka, "Plagiarism detection across distant language pairs," in *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 37–45, Beijing, China, August 2010.

[13] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis," in *Proceedings of the 20th international joint conference on Artifical intelligence*, pp. 1606–1611, Hyderabad India, January 2007.

[14] J. Ferrero, F. Agnes, L. Besacier, and D. Schwab, "Using word embedding for cross-language plagiarism detection," 2017, https://arxiv.org/abs/1702.03082.

[15] M. Kusner, Y. Sun, N. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *Proceedings of the International Conference on Machine Learning*, pp. 957–966, Lille France, July 2015.

[16] M. Guo, Q. Shen, Y. Yang, H. Ge, D. Cer, and K. Stevens, "Effective parallel corpus mining using bilingual sentence embeddings," 2018, https://arxiv.org/abs/1807.11906.

[17] H. Zhang, "A comparative algorithm of the similarity in "blank" concept for Chinese and western poetics in the context of Internet," in *Proceedings of the 3rd International Conference on Artificial Intelligence and Advanced Manufacture*, pp. 1778–1782, Manchester, UK, 2021.

[18] W. L. Zhuang and E. Chang, "Neobility at SemEval-2017 task 1: An attention-based sentence similarity model," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 164–169, Association for Computational Linguistics, Vancouver, Canada, August 2017.