A novel sentence similarity model with word embedding based on convolutional neural network: Sentence Similarity Model with Word Embedding based on Convolutional Neural Network

Article *in* Concurrency and Computation Practice and Experience · January 2018

DOI: 10.1002/cpe.4415

CITATIONS
13

READS
1,215

3 authors, including:

Haipeng Yao
Beijing University of Posts and Telecommunications
142 PUBLICATIONS **1,269** CITATIONS
SEE PROFILE

Peiying Zhang
Beijing University of Posts and Telecommunications
82 PUBLICATIONS **602** CITATIONS
SEE PROFILE

Received: 7 November 2017 | Revised: 28 November 2017 | Accepted: 30 November 2017
DOI: 10.1002/cpe.4415

SPECIAL ISSUE PAPER

WILEY

A novel sentence similarity model with word embedding based on convolutional neural network

Haipeng Yao[1] | Huiwen Liu[1] | Peiying Zhang[1,2]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
[2]College of Computer and Communication Engineering, China University of Petroleum, Qingdao, China

Correspondence
Haipeng Yao, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China.
Email: yaohaipeng@bupt.edu.cn

Present Address
Haipeng Yao, Haidian District Xitucheng Road 10, Beijing 100876, China.

Funding information
National Natural Science Foundation of China, Grant/Award Number: 61471056; Shandong Provincial Natural Science Foundation of China, Grant/Award Number: ZR2014FQ018; BUPT-SICE Excellent Graduate Students Innovation Fund

Summary

In this paper, we propose an effective model for the similarity metrics of English sentences. In the model, we first make use of word embedding and convolutional neural network (CNN) to produce a sentence vector and then leverage the information of the sentence vector pair to calculate the score of sentence similarity. Considering the case of long-range semantic dependencies between words, we propose a novel method transforming word embeddings to construct the three-dimensional sentence feature tensor. In addition, we incorporate the k-max pooling into the convolutional neural network to adapt to variable lengths of input sentences. The proposed model requires no external resource such as WordNet and parse tree. Meanwhile, it consumes very little time for training. Finally, we carried out extensive simulations to evaluate the performance of our model compared with other state-of-the-art works. Experimental results on SemEval 2014 task (SICK test corpus) indicated that our model can achieve a good performance in the terms of Pearson correlation coefficient, Spearman correlation coefficient, and mean squared errors. Furthermore, experimental results on Microsoft research paraphrase identification (MSRP) indicated that our model can achieve an excellent performance in the terms of F1 and Accuracy.

KEYWORDS
convolutional neural network, sentence similarity, word embedding

1 | INTRODUCTION

Sentence similarity measurement is a fundamental problem in the research of natural language processing (NLP) and has been used in many language processing tasks such as query ranking, question answering, paraphrase identification, and paraphrase generation, to cite a few. Understanding the similarity between sentences is not a simple task for machines because both semantic information and syntactic information should be taken into consideration. The distributed word embedding has been explicitly encoded many linguistic regularities and patterns in the works of Mikolov et al[1,2] and others, and it has been widely utilized to extract semantic features of sentences. The syntax parsing is a challenge task due to its complexity; therefore, it is a good idea that makes machines hiddenly learn grammar rules from corpus. Recently, as the increasing interest in neural networks, the idea is realized by means of neural network-based models.

In 2008, Collobert and Wetson[3] proposed a convolutional neural network-based architecture for modeling sentences and used it in several learning tasks such as named entity tags, semantic roles, and semantically similar words. In their studies, words were embedded into the multi-dimension space. Inspired by their works, other researchers utilized word embedding and convolutional neural network in paraphrase identification,[4,5] sentence classification,[6-8] and semantic similarity measurement.[6,9,10] Recurrent neural network, a time sequence neural network with variable-length input, is good choice for sequence modeling tasks like modeling sentences. Socher et al[11] applied RNN in their task that finding and describing images with sentences. The disadvantage of RNN is its gradients vanishing over long sequence. To avoid the vanishing gradient problem, the long short-term memory networks (LSTM) were proposed, which is better in learning long-range dependencies. Tai et al[6] and Mueller and Thyagarajan[12] used LSTM for sentence similarity measurement.

However, both long short-term memory networks and deep convolutional neural networks have a disadvantage of time-consuming when they are utilized for semantic similarity measurement. In addition, deep convolutional neural networks need to be modified for measuring sentence similarity. The structures of them universally are complicated. Since the measurement of sentence similarity is a fundamental task in NLP and more commonly serves as a subtask that aids in solving other larger tasks, these models that have complicated structure and cost long time for training will consume extra resources that assigned for other subtasks and slow down the proceeding of the whole task.

In this paper, we propose a novel model with word embedding and convolutional neural network for sentence similarity measurement. The major contributions of this paper are as follows.

1. To reduce the training time and simplify architecture structure, we propose a novel sentence similarity model that is different from traditional long short-term memory networks and deep convolutional neural networks. The main body of our model is a shallow convolutional neural network without requiring fully connected layer meanwhile modifying its structure targeting at the task of sentence similarity measurement.
2. The sentence with variable length is embedded into a multiple dimensional space. We extend the method of word embedding from word-level to sentence-level, with the aim of putting word similarity computation methods into sentence similarity measurement. Additionally, our model can handle any length sentences without needing clipping and padding operations.
3. We evaluate our model using different evaluation metrics with two different kinds of tasks, namely, semantic relatedness task (SemEval 2014, Task 1) and the Microsoft research paraphrase identification task. The obtained results achieve a good performance on both tasks.

The remainder of our paper is organized as follows. Firstly, we introduce the related studies that apply neural networks to semantic similarity measurement in Section 2. Secondly, we depict the details of our sentence similarity computation model in Section 3. Thirdly, our experiments are presented in Section 4. Finally, we draw the conclusion in Section 5.

2 | RELATED WORKS

There are plenty of sentence similarity computation methods in the field of natural language processing. These computation methods of sentence similarity can be roughly categorized into three aspects according to the type of the applied neural networks, including CNN, RNN, and DNN. This section reviews the related works from the three aspects.

2.1 | The computation methods of sentence similarity based on CNN

Kalchbrenner et al[7] proposed a dynamic convolutional neural network (DCNN) model for modeling sentences and carried out four experiments to validate the proposed model. The proposed sentence representation model is characterized by three aspects including dynamic k-max pooling, requiring no external resources and language-independent. He et al[9] introduced a sentence model based on convolutional neural network with the purpose of extracting features from different perspectives and making a comparison of sentence representations by means of different similarity metrics at different granularities. To shorten the training time, He and Lin[10] leveraged the combination of convolutional neural network and bidirectional long short-term memory network to measure the textual similarity and achieved satisfactory results on three SemEval tasks and two answer selection tasks.

He et al[9] and He and Lin[10] used the deep neural network to calculate the sentence similarity measurement. Particularly, He and Lin applied a deep architecture with 19 sublayers' CNN and bidirectional LSTM. However, our model is different from their model in the architecture. Firstly, we propose a new method to construct the input map of the network. Secondly, we apply a shallow convolutional neural network and we make some modification to the network. Thirdly, the output of our neural network is a vector that uses to compute the score of sentence similarity. In other works, the score of sentence similarity is directly computed by the neural network.

2.2 | The computation methods of sentence similarity based on RNN

Socher et al[13] proposed an unsupervised recursive auto encoder (RAE) for paraphrase detection, which is based on an unfolding objective and learn to obtain feature vectors for phrases in syntactic trees. These extracted features are applied to compute the word-wise and phrase-wise similarity from a sentence pair. Hermann and Blunsom[14] introduced a model based on combinatory categorical auto encoders, which guide a non-linear transformation of meaning within a sentence and learn high dimensional embeddings for sentences and demonstrated that the representations of sentences learned from the model are both effective and general in a range of tasks. Tai et al[6] proposed the tree-structured long short-term memory networks (Tree-LSTMs) for modeling sentences, which composes its states from an input vector and the hidden states of arbitrarily many child units. The Tree-LSTM was evaluated on tasks of semantic relatedness prediction and sentiment classification, and it had good performance on both tasks.

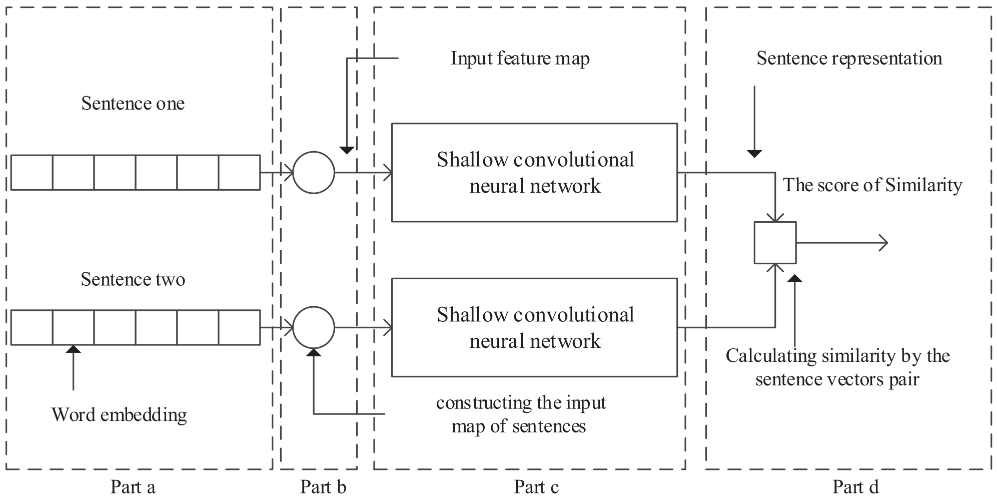
Sentence one
Sentence two
Word embedding
Input feature map
Shallow convolutional neural network
Shallow convolutional neural network
constructing the input map of sentences
Sentence representation
The score of Similarity
Calculating similarity by the sentence vectors pair
Part a
Part b
Part c
Part d

FIGURE 1 The architecture of the model

2.3 | The computation methods of sentence similarity based on DNN

Lu and Li[15] developed a deep neural network for matching short texts, which can effectively capture the nonlinearity and hierarchy in the matching texts. First, the authors considered interactions between components within the two texts and construct an interaction space. Then, they input the local decisions in the space into the corresponding neurons of the deep neural network. Finally, they used the logistic regression at the output layer of the deep neural network to calculate the final matching score. Their work was superior to various inner-product based matching models.

3 | THE PROPOSED MODEL WSV-SCNN-SV

In this section, we propose our neural network-based model, namely, "the word set vectors-the shallow convolutional neural network-the sentence vector" (WSV-SCNN-SV). The main body of the model is the shallow convolutional neural network, which takes the group of "word set vectors" as the input feature map and outputs the sentence representation—the sentence vector. The "word set vector" is a new idea considering as the improvement of the word embedding. Compared with word embedding, it contains more sematic information of sentences. We compute the similarity based on the sentence vectors that learned by the convolutional neural network instead of employing the convolutional neural network.

The organization of this section is summarized as follows. First, we introduce the whole architecture of our model, and then, we elaborate on each part of our model in detail.

3.1 | The overall architecture of the model

The hybrid architecture of WSV-SCNN-SV is shown in Figure 1. In the architecture, there are four parts.

Part a. Sentences pre-processing: Since the raw sentence is not accepted by machines, it should be processed and encoded before it input into machines. The pre-process of the sentences includes word segmentation, removal of stop words, stemming, and so on. In our work, we use word embedding to encode sentence in the way that words from each sentence are represented by their corresponding word vector after the pre-processing.

Part b. The input model: We propose a novel method that transforms the sequence of word vectors into a three-dimensional tensor. The tensor that contains rich semantic and syntactic information of the sentence is the input feature map of the next convolutional neural network.

Part c. Our shallow convolutional neural network: The work of our convolutional neural network is learning the semantic and syntactic features from the input tensor and producing the sentence representation. We make some modification to the convolutional neural network. First, we remove fully connected layer from our model. After flowing over several convolutional and pooling layers, the tensor is converted into a vector that extracts rich features from the input tensor. Second, we apply both k-max pooling and max pooling operation in the model. Third, we make our convolutional neural network support the input feature map with unfixed size by means of k-max pooling operation.

Part d. Similarity computation: The sentence vector pair is used to compute the score of similarity. Many methods are available for similarity measurement between two vectors. In our work, we use cosine distance, Euclidean distance, and Manhattan distance, respectively.

In summary, the aim of the first three parts of the model is to accomplish the task that embeds sentences into a high dimensional space. The goal of the last part of our model is to calculate the score of similarity with the sentence vector pair.

3.2 | Sentences pre-processing

Since each word is an atomic semantic unit for a sentence, it is of great significance for machines to capture fine-grained features from sentences. Different sentences may use different words to convey the high similar information, for instance, "He studies computer science in college" and "His major is CS". Many remarkable research studies have been done in modeling words like distributed word embedding, which explicitly encode many linguistic regularities and word similarities.

One of notable works on distributed word embedding is word2vec model.[1,2] In word2vec model, the result of a vector calculation vector("Madrid") – vector("Spain") + vector("France") is closer to vector("Paris") than to any other word vector. It is universal in related papers that base on word2vec to model sentence measuring similarity. Similarly, in this paper, we apply pre-trained word embedding to model words.

There are many simple methods to produce the sentence representation using word embedding. Assume $\mathbf{w}_i = (w_i^1, w_i^2, \dots, w_i^d)$ is a d-dimensional word vector corresponding to the i-th word in the sentence. A sentence contained n words can be represented as follows:

$$S = \frac{1}{n}\sum_{i=1}^{n}\mathbf{w}_i \tag{1}$$

$$S = \left(\mathbf{w}_1^T, \mathbf{w}_2^T, \dots, \mathbf{w}_n^T\right). \tag{2}$$

In prior studies, these sentence representations are input into neural network to finish final task. Formula 1 uses the average value of these word vectors to express the sentence,[16-21] despite this process is very simple, its performance in practice is not poor. Formula 2 concatenates word vectors to produce a two-dimensional matrix to express the sentence, which is a common method to represent sentence. Different from formula 1, the method of formula 2 keeps order information of words. Kim[8] used the method, and Hu et al[4] made some improvement of formula 2.

However, our idea is different from the previous work. Our motivation is that sentence representation should learn from neural network as the word embedding does. After words in the sentence are represented by their corresponding word vector, we construct a three-dimensional tensor and then put the tensor into the convolutional neural network to learn the sentence representation.

3.3 | The input model based on "binary-gram word set vector"

The word is not the only factor to be taken into consideration in the sentence representation. The dependencies between word and phrase and between phrase and phrase cannot be ignored. The representation of the sentence in formula 2 can be treated as a feature map with one width, n height, and d channels. However, the disadvantage of the representation is that long-range semantic dependencies between words have been ignored.

To consider the long-range semantic dependencies in our model, we propose a new vector dubbed "word set vector". The idea of word set vector is inspired by the idea that sentences can be represented by the average of word vectors. We extend this idea from sentences to phrases and collections of words that are semantic dependencies with each other. Word set vector $\overline{\mathbf{w}}$ composed with j word vectors $\mathbf{w}_{l_1}, \mathbf{w}_{l_2}, \dots, \mathbf{w}_{l_j}$ is defined as follows:

$$\overline{\mathbf{w}} = \sum_{i=1}^{j}\lambda_i \mathbf{w}_{l_i}, \tag{3}$$

where the sequence $\{l_1, l_2, \dots, l_j\}$ denotes a subsequence of the sequence $\{1, 2, \dots, n\}$, λ denotes the weight of word vector ($\sum_{i=1}^{j} \lambda_i = 1, \lambda_i > 0$).

The word set vector composed by word vectors $\mathbf{w}_{l_1}, \mathbf{w}_{l_2}, \dots, \mathbf{w}_{l_j}$ is order sensitive. The word set vector $\mathbf{w}$ composed with j word vectors (allowed repeated word vectors) is defined as j-gram word set vector. We deem that the parameter λ should be trained by machines, it can be calculated as follows:

$$\lambda_i = \frac{\exp(\alpha_i)}{\sum_j \exp(\alpha_j)}, \alpha_j \epsilon \mathbb{R}. \tag{4}$$

The word set vector uses the weight λ to punish disorder word vectors. A sentence with n words has n^j j-gram word set vectors. A sentence with n words has n^2 d-dimensional binary-gram word set vectors, which can be transformed into a feature map with n width, n height, and d channels. The binary-gram word set vector is calculated by

$$\widetilde{\mathbf{w}}_{l_1 l_2} = \lambda_1 \mathbf{w}_{l_1} + \lambda_2 \mathbf{w}_{l_2}, \tag{5}$$

where $l_1, l_2 \in \{1, 2, \dots, \mathrm{n}\}, \lambda_1 + \lambda_2 = 1, \lambda_1, \lambda_2 > 0$. If $l_1 = l_2$, $\widetilde{\mathbf{w}}_{l_1 l_2}$ denotes a binary-gram word set vector. The weight λ_1 and λ_2 are calculated by gradient descent.

Algorithm 1 shows how the input feature map is composed. The units of the map are these binary-gram word set vectors in the sentence.

Algorithm 1 The computation process of the input feature map

```
Input:  Word embeddings, $w_1, w_2, \dots, w_n$
Output:  The input feature map, map
 1: Initialize the input feature map: map $\in \mathbb{R}^{n \times n \times d}$ to all zero
 2: Initialize variables: $\alpha_1, \alpha_2$ to 0.4, 0.6
 3: $\lambda_1 = \exp(\alpha_1)/(\exp(\alpha_1) + \exp(\alpha_2))$
 4: $\lambda_2 = \exp(\alpha_2)/(\exp(\alpha_1) + \exp(\alpha_2))$
 5: for $i = 1$ to $n$ do
 6:     for $j = 1$ to $n$ do
 7:         for depth = 1 to $d$ do
 8:             map[$i$][$j$][depth] = $\lambda_1 \cdot w_i$[depth] + $\lambda_2 \cdot w_j$[depth]
 9:         end for
10:     end for
11: end for
```

Figure 2 illustrates the feature map that is composed of n^2 word set vectors. It is one of the creative works in this paper that extending the feature map of sentences from a two-dimensional matrix to a three-dimensional tensor. As the feature map is the same as the image data in form, we hope that the next convolutional neural network can fully extract sentence features from the input feature map as it does on images.

3.4 | Our shallow convolutional neural network

The complete multi-layer's convolutional neural network should contain fully connected layer just like Figure 3A. In our convolutional neural network, we remove the fully connected layer from the network as it is not essential for learning sentence vectors and consumes time for training. Our proposed convolutional neural network only contains layers that consist of convolutional sublayers and pooling sublayers. In addition, a sum function that computes the sentence vectors is applied at the output of the last layer.

Convolutional sublayers: The formula of convolution is as follows:

$$X_j^l = \phi\left(\sum_{i \in M_j} X_i^{l-1} * K_{ij}^l + b_j^l\right), \tag{6}$$

where X_j^l denotes the j-th map on the l-th sublayer, $\phi(x)$ denotes the active function, M_j denotes the number of maps on the $(l-1)$-th sublayer, K^l denotes the filter of the l-th sublayer, and b_j^l denotes the bias.

We do not make any modification to convolutional sublayers. As can be seen in Figure 3A, zero padding (dashed blocks are zero padding) is used in the convolutional operation so that the size of input feature map of convolutional layer is the same as the size of output feature map of the convolutional layer.

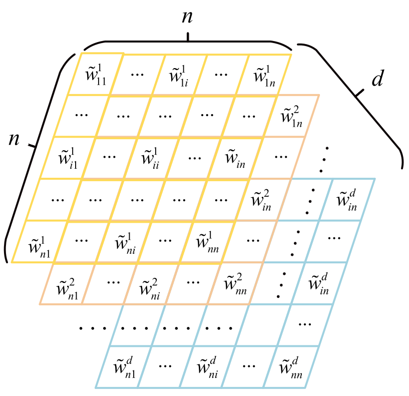

FIGURE 2 The input feature map of the convolutional neural network where $\tilde{w}_{ij}^k = \lambda_1 w_i^k + \lambda_2 w_j^k$ denotes the k-th dimensional value of word set vector that is composed of word vector $\mathbf{w}_i$ and $\mathbf{w}_j$

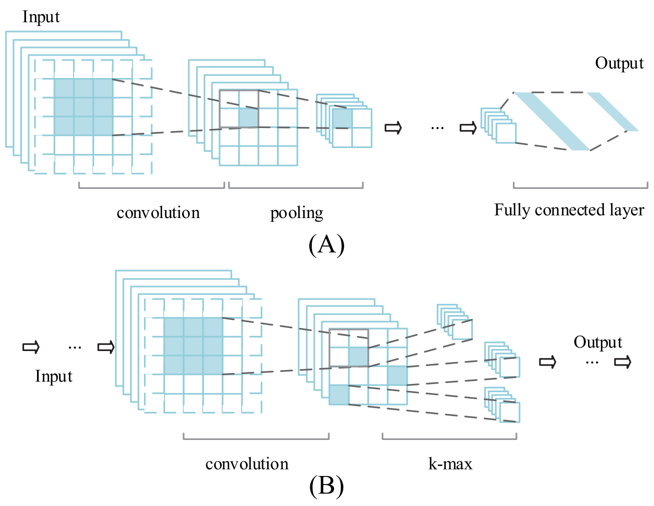

FIGURE 3 The convolutional neural network, k-max pooling. A, The convolutional neural network with max-pooling operation; B, The convolutional neural network with k-max pooling operation

The activate function of convolutional sublayers is ReLU function

$$\phi(x) = \begin{cases} x \ \text{ if } \ x \geq 0 \\ 0 \ \text{ if } \ x < 0. \end{cases} \tag{7}$$

Although, the performance of ReLU function is poor than sigmoid function's, it is simple and takes less training time to achieve convergence.

Max pooling sublayers: Pooling sublayers are followed convolutional sublayers. The pooling operation is also called down sampling. In pooling operation, both width dimension and height dimension of the feature map will be reduced. However, the depth dimension of the feature map remains unchanged. Taking 2×2 pooling window for example (it can be seen in Figure 3A), the width and height of feature map will reduce half after a pooling operation. If the size of input feature map of multi-layer network is defined, after several times pooling operation, the size of output feature map is also definite.

The formula of convolution is as follows:

$$X_j^l = \beta_j^l \text{down}\left(X_j^{l-1}\right) + \text{bias}_j^l, \tag{8}$$

wh ere β_j^l denotes the weight, down(x) denotes the down sampling, and bias_j^l denotes the bias.

The k-max pooling sublayer: The length of the sentence is variable. To support sentences with different length, the size of input layer of the convolution neural network must be changeable. To remain the same size of output layer of the convolutional neural network with variable input feature map, k-max pooling is used on the last pooling layer to replace max pooling operation. Figure 3B shows k-max pooling operation ($k = 3$) on the input feature map. The pseudo code of k-max pooling is as follows.

Algorithm 2 The computation process of the k-max pooling

```
Input: The three-dimensional feature map, tensor $\in \mathbb{R}^{\text{width}\times\text{height}\times\text{depth}}$
Output: The $k$ vectors, vectors $\in \mathbb{R}^{k\times\text{depth}}$;
1:  function KMAX(tensor)
2:      tensor = reshape(tensor, [width × height, depth])
3:      //After the reshape operation, tensor is a two-dimensional matrix: tensor $\in \mathbb{R}^{(\text{width}\times\text{height})\times\text{depth}}$
4:      for $i = 1$ to tensor.depth do
5:          tensor[:][$i$]=AscendingSort(tensor[:][$i$])
6:      end for
7:      for $i = 1$ to k do
8:          vectors[$i$][:] = tensor[$i$][:]
9:      end for
10:     return vectors
11: end function
```

The k-max pooling used in our model is different from the k-max pooling used in the work of Kalchbrenner et al[7] In the aforementioned work,[7] after k-max pooling, the relative position of data in the original feature map is preserved. In contrast, we discard the relative position of data because we subsequently summed the data up to calculate the sentence vector. In Figure 3B, the k-max operation on any $n \times n \times d$ ($n \geq 2$) input feature map produces three d-dimensional vectors.

The sum function: Taking the sentence "The cat sits on the mat" for example, its subject, predicate, and object convey enough meaning to imply the whole sentence. Similarly, we hope that words or phrases that convey the meaning of sentence can be "extracted" from the sentence by means of k-max pooling operation. The k vectors outputted by k-max pooling is called the vector group of the sentence. There is no fully connected layer in our network. The sentence vector is the average value of the vector group of the sentence and is calculated by

$$v_S = \frac{1}{k}\sum_{i=1}^{k} u_i, \tag{9}$$

where v_S represents the sentence vector and $u_1, \dots, u_i, \dots, u_k$ denotes the vector group of the sentence.

Algorithm 3 shows the pseudo code of the shallow convolutional neural network.

Algorithm 3 The computation process of the shallow convolutional neural network

```
Input: The input feature map, $map_0$
Output: The sentence vector, $v_s$
1: Initialize the filter: $filter^1, filter^2, \dots, filter^l$
2: for $i = 1$ to $l - 1$ do
3:     $conv_i = ReLu(convolute(map_{i-1}, filter^i, padding = 'same'))$
4:     $map_i = MaxPooling(conv_i)$
5: end for
6: $conv_l = ReLu(convolute(map_{l-1}, filter^l, padding = 'same'))$
7: $vectors = kMax(conv_l)$
8: $v_s = average(vectors[1], vectors[2], \dots, vectors[k])$
```

In our application, the number of layers in our network is no more than three. The work of fully connected layers is substituted by the next calculation unit. Simple in structure and fast in training are the advantages of our model. The model with two layers of network is shown in Figure 4.

3.5 | Similarity computation

The similarity of sentences is calculated by the sentence vector pair. There are many methods to compute the similarity of vectors. We choose cosine distance, Euclidean distance, and Manhattan distance to evaluate the score of similarity. However, the value range of Euclidean distance and

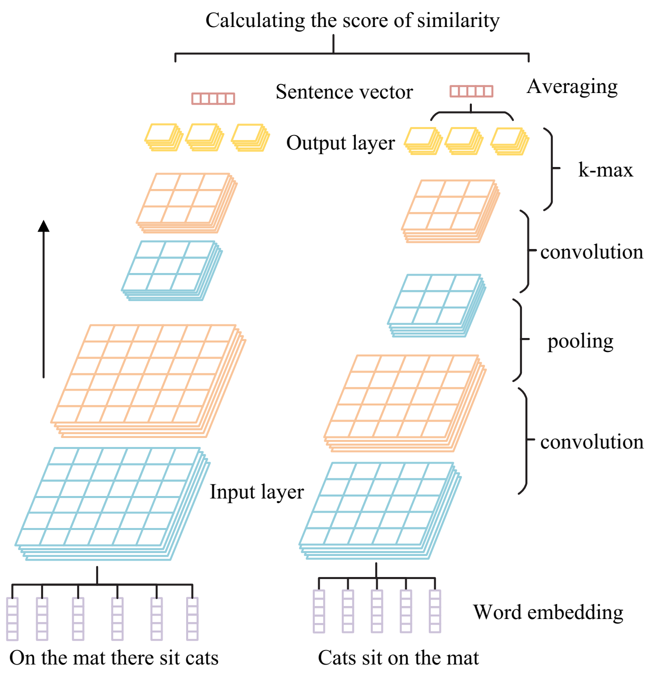

FIGURE 4 The sentence similarity model

Manhattan distance is not [0, 1]. We need to make modification to them. The score calculated by Euclidean distance and Manhattan distance is as follows:

$$\text{score} = f\left(||v_{s1} - v_{s2}||\right), \text{score} \in [0, 1], \tag{10}$$

where *f* denotes a monotone decreasing function, and the codomain of *f* is [0,1]. We use three different *f* in the model: $f_1(x) = e^{-x}$, $f_2(x) = \frac{1}{1+e^{-x}}$, and $f_3(x) = \frac{1}{1+x}$. Since the output of ReLu function is not negative, there is no need to concern the negative value of cosine distance. The cosine distance of the sentences vector pair can be directly used as the score of similarity.

4 | EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we present detailed experimental setup and loss functions in our training process, after which experimental analysis is illustrated respectively. Two different data sets SICK and MSRP are used to evaluate our model.

4.1 | Experimental setup

We did all the experiments on a personal computer with 8 gigabytes memory and Intel i7 quad core CPU. In experiments, we use GloVe word embedding[22] (trained on Wikipedia 2014 + Gigaword 5), and we fine-tune the word embeddings on training. Words that are not present in the GloVe are initialized randomly.

If padding is not allowed in pooling sublayer and k-max pooling sublayer, the length of sentence must meet a lower limit. The length of the sentence satisfies

$$\text{Length}_S \geq (f_w)^{L-1}k, \tag{11}$$

where f_w represents the width of pooling window (in this paper, the width of pooling window is equal to its height), *k* denotes the parameter of k-max, and *L* denotes the number of hidden layers.

In the model, the number of channels of input layer is the same as the dimension of word embedding (commonly more than 50-dimension). Adding a new hidden layer to network will greatly increase the number of training parameters in the model. To limit the number of parameters, the number of layers *L* is no more than three.

On training task, to keep the same size of input map in the same batch, we have to use zero padding in input layer. However, on testing task, the model supports the different size of input map.

The model contains many hyper parameters. This is a shortcoming of the model. There are the number of layers *L*, the parameter *k*, the width of convolutional filter, the width of pooling window, and the number of hidden feature maps to be artificially specified. But the model with non-fully connected layer and fewer hidden layers is less than most of deep neural networks on the total amount of parameters.

4.2 | Training

The data sets of our experiments are SemEval-2014 Sentences Involving Compositional Knowledge (SICK) data and Microsoft research paraphrase identification (MSRP).[23,24] SemEval (Semantic Evaluation) is a computational semantic analysis system organized by the Special Interest Group on the Lexicon of the Association for Computational Linguistics. SICK is the data set of SemEval-2014 Task 1 competition,[25] the aim of which is evaluated compositional distributional semantic models on full sentences through semantic relatedness and entailment. Many participants submitted their works on the task and their results are available in the work of Marelli et al[26] SICK contains training data (4500 sentence pairs), trial data (500 sentence pairs), and test data (4927 sentence pairs). Relatedness score of sentence pair can range from 1 (completely unrelated) to 5 (very related). MSRP contains training data (4076 sentence pairs) and test data (1725 sentence pairs). Relatedness score of sentence pair is one or zero.

Hyper parameters of the model were set as follows: parameter k of k-max pooling was 3; the size of convolutional filter was 3 × 3; and the single training batch size was set to be 50. We are fine tuning the word embedding on both tasks. The loss function of these two tasks is different.

On the SICK data set, we train model to minimize the loss function of mean squared errors (MSE)

$$\text{loss}_1 = \frac{1}{m}\sum_{i=1}^{m}(\text{sim}_p - \text{sim}_l)^2, \tag{12}$$

where sim_p denotes the predicted value of similarity score, sim_l represents similarity score that is calculated as the average of ten human ratings collected for each pair, and *m* is the scale of training data set.

Besides, to explore the influence of loss function on the result, we use the other loss function, ie, KL divergence loss

$$\text{loss}_2 = \frac{1}{m}\sum_{i=1}^{m}\left[q \cdot \log\frac{q}{p} + (1-q) \cdot \log\frac{(1-q)}{(1-p)}\right], \tag{13}$$

where p is normalized sim_p and q is normalized sim_l. Considering denominator p and $1 - p$ could be zero, Laplace Smoothing is being used on this loss function.

On the MSRP data set, we train model to minimize cross entropy between the predicted value and the label.

4.3 | Experiments analysis

Figure 5 illustrates the results on the SICK test set. The six results at the bottom of the Figure are the best six results released by SemEval. These results represent the high level of traditional methods. It can be seen that the result of this model is ranked third in the aforementioned results, according to Pearson coefficient (SemEval official ranking). Similarly, according to Spearman coefficient, the result is ranked the third, and the result on MSE is ranked the fourth, very closed to the third 0.3593. There is no distinct difference between the result of our model and the best results. Differences between our result with the best on Pearson, Spearman, and MSE respectively are 0.0159, 0.0137, and 0.0382. Figure 5 also provides a comparison of the result of this work with the results of two different RNN models. Compared with RNN, it is disadvantaged for the CNN network to deal with long-term and short-term dependencies between words in the sentence. Surprisingly, the test result of the model is slightly better than the results of these two RNN models. But the results of our model are not as good as the results of literatures (see the works of He et al[9] and He and Lin[10]). The results of their work on Pearson coefficient are 0.8686 and 0.8784, respectively. Despite the performance of our model is poorer than these deep neural networks, our work is still valuable: compared with traditional methods, our work achieves better performance, and compared with deep neural networks, our model is very practical in applications since it is simple in architecture and fast in the training.

Table 1 presents the methods and resources used by several models that well-performed on the SICK data set. It can be seen that the well-performed ECNU model uses four learning methods, WordNet, and additional Corpus; The second ranked model, ie, The Meaning Factory, also uses three different resources including WordNet. In contrast, our model only uses word embeddings and convolution neural networks. Our model is much simpler in structure but can achieve the same degree of test performance.

Figure 6 demonstrates the training curve of our model on SICK set (Learning rate is 10^{-3}). From Figure 6, It can be seen that the performance of these two loss functions has no significant difference on the test results. The results show that both curves are fast convergence.

We used different dimensional word embeddings to test our model and recorded their training time. We chose a single-layer network to accomplish the experiments. Table 2 shows the corresponding results. All of experiments that list the training time are done on the same computer.

As can be seen from Table 2, the result of a single-layer network which contains 1.62 million parameters and spent around 80 minutes on training can rank the third in Figure 5. After reducing the half number of parameters, the test result does not show obvious difference with the same epoches. The Pearson correlation coefficient of the result reaches 0.793, the fourth level in Figure 5, and is very closed to that of the third. Meanwhile, the consuming time can be reduced to a half: the training process can be accomplished in 37 minutes.

Although the model of the third line in the Table (100-dimensional word vector) has not bright performance under ten epoches, if the epoches are 40, the metrics of its Pearson correlation coefficient was 0.7868 around, its mean squared error was approximately 0.3878, while the parameters of the whole model are only 3/8 of the second case in the Table.

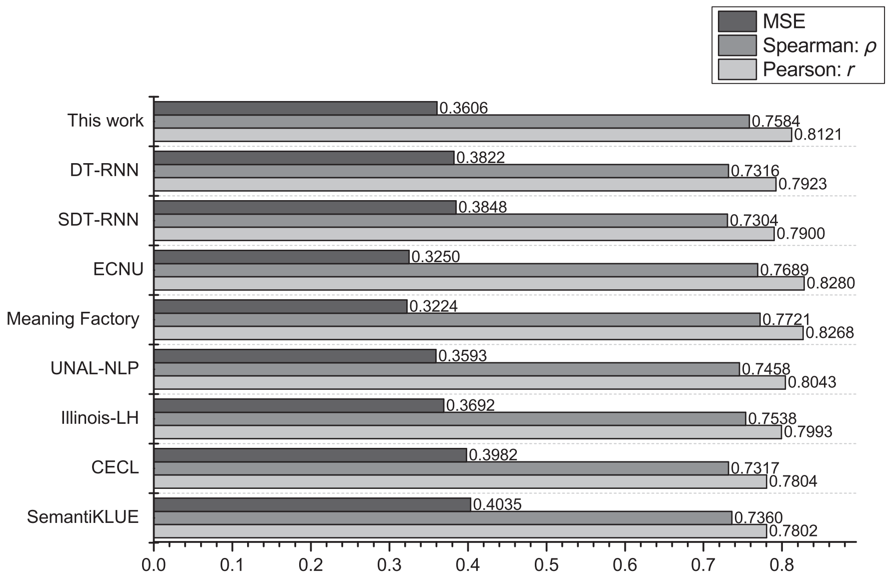

FIGURE 5 The test result on SICK data set. DT-RNN,[11] SDT-RNN,[11] ECNU,[27] Meaning Factory,[28] UNAL-NLP,[29] Illinois-LH,[30] CECL,[31] and SemantiKLUE[32]

TABLE 1 The methods and resources that models applied[26]

Model	SVM and Kernel methods	K-Nearest Neighbors	Classifier Combination	Random Forest	Convolutional Neural Network	Other	WordNet	Paraphrases DB	Other Corpora	ImageFlicker	STS MSR-Video Descriotion	Word Embeddings
	Learning Methods						**External Resources**					
ECNU	Y	Y	Y	Y			Y		Y			
The Meaning Factory				Y			Y	Y	Y			
UNAL-NLP						Y	Y	Y	Y			
Illinois-LH						Y	Y			Y	Y	
This work					Y							Y

Y represents the model used this method/resource

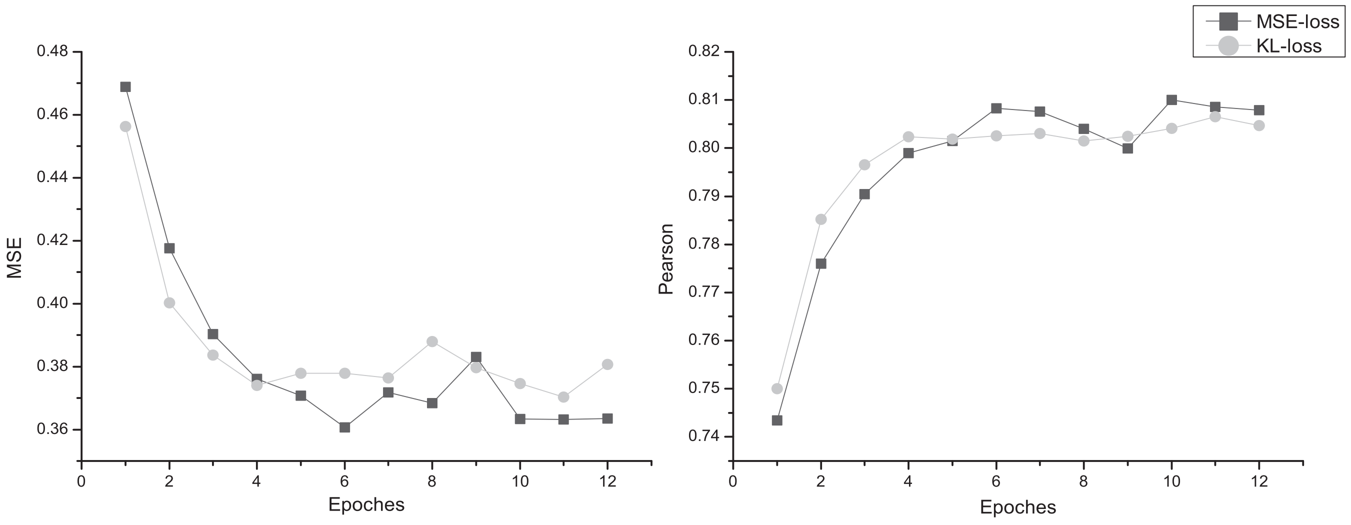

FIGURE 6 The relatedness curve of mean square errors, Pearson coefficient, and epoches on test data set using KL divergence loss function and MSE loss function

TABLE 2 The results on the different settings of the model

params	r	ρ	MSE	T
1620002 (300)	0.8069	0.7433	0.3897	4731
720002 (200)	0.7930	0.7308	0.4269	2197
270002 (100)	0.7679	0.7052	0.4419	819
90002 (50)	0.7623	0.6999	0.5076	495

The "params" represents the parameters of the whole model to be trained, the content in the parentheses denotes the dimension of word vector, r denotes Pearson correlation coefficient, ρ denotes Spearman correlation coefficient, MSE denotes mean square error, and T denotes the average time spent for the training (Unit: seconds). The epoches of the results are ten.

Figure 7 shows that the result of our model that is pre-trained on SICK corpus outperform others on MSRP test set. Meanwhile, the result of our model without pre-training does not perform well. Thus, pre-training is helpful to improve the performance of our model. We assume that our model may perform better after pre-training on a large amounts of appropriate training data, which we leave to our future work. (In Figure 7, the work of Hu et al[4] is based on CNN, other works in Figure 7 are based on traditional methods. The results of Baseline and the work of Rus et al[33] are reported by Hu et al,[4] and the result of the work of Rus et al[33] is the best one in the report.)

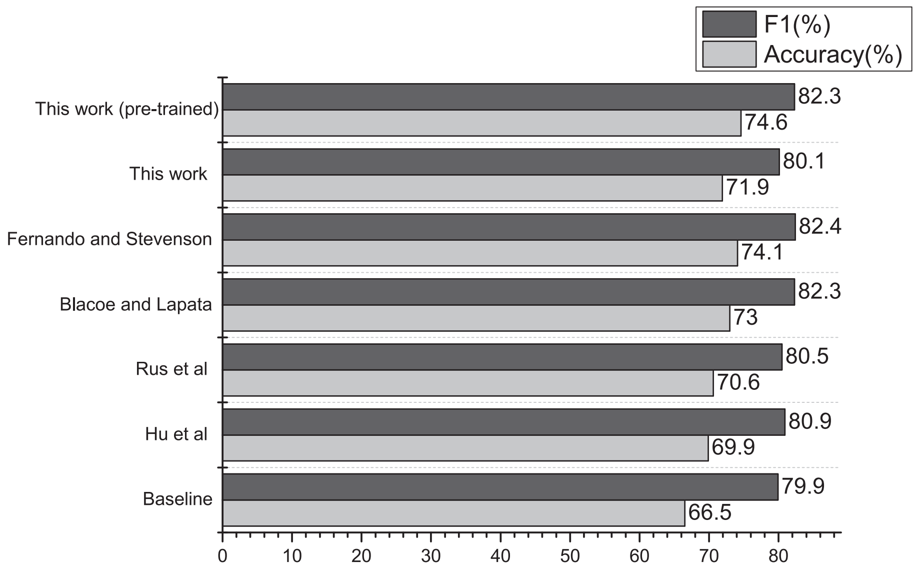

FIGURE 7 The test result on MSRP data set. Baseline,[4] Hu et al,[4] Rus et al,[33] Blacoe and Lapata,[34] and Fernando and Stevenson[35]

5 | CONCLUSION

This paper presents a neural network-based model with word embedding to measure the similarity of sentences. In this field, many complex architectures are proposed to measure the sentence similarity. These models do not perform well on the reduction of resource occupation. The model of this paper is proposed to deal with this type of problem.

In this paper, the number of layers of the model is less than four. The simple model has less parameters and costs short time for training. Our experiments proved that even the shallow convolutional neural network can also achieve a good prediction of the score of similarity. This paper also compares the influence of the different dimensions of Word Embeddings and the different loss function on the test results. In experiments, it is proved that our model is very flexible. The increase in the dimensions of the word embedding can reach the better effect, and the decrease in the dimensions of the word embedding will greatly improve the training speed while reduces a little on Pearson correlation coefficient.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (grant 61471056), the Shandong Provincial Natural Science Foundation of China (grant ZR2014FQ018), and the BUPT-SICE Excellent Graduate Students Innovation Fund. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

ORCID

Haipeng Yao http://orcid.org/0000-0003-1391-7363

REFERENCES

1. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ, eds. *Advances in Neural Information Processing Systems 26*. New York, NY: Curran Associates, Inc; 2013:3111-3119.
2. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. Computation and Language; 2013.
3. Collobert R, Wetson J. A unified architecture for natural language processing: Deep neural networks with multitask learning. Paper presented at: International Conference on Machine Learning; 2008; Helsinki, Finland.
4. Hu B, Lu Z, Li H, Chen Q. Convolutional neural network architectures for matching natural language sentences. Paper presented at: International Conference on Neural Information Processing Systems; 2014; Montreal, Canada.
5. Yin W, Schütze H. Convolutional neural network for paraphrase identification. Paper presented at: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2015; Denver, CO.
6. Tai KS, Socher R, Manning CD. Improved semantic representations from tree-structured long short-term memory networks. *Comput Sci*. 2015;5(1):36.

7. Kalchbrenner N, Grefenstette E, Blunsom P. A convolutional neural network for modelling sentences. Paper presented at: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics; 2014; Baltimore, MD.
8. Kim Y. Convolutional neural networks for sentence classification. Paper presented at: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2014; Doha, Qatar.
9. He H, Gimpel K, Lin J. Multi-perspective sentence similarity modeling with convolutional neural networks. Paper presented at: Conference on Empirical Methods in Natural Language Processing; 2015; Lisbon, Portugal.
10. He H, Lin J. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. Paper presented at: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2016; San Diego, CA.
11. Socher R, Karpathy A, Le QV, Manning CD, Ng AY. Grounded compositional semantics for finding and describing images with sentences. 2013. https://nlp.stanford.edu
12. Mueller J, Thyagarajan A. Siamese recurrent architectures for learning sentence similarity. Paper presented at: Thirtieth AAAI Conference on Artificial Intelligence; 2016; Phoenix, AZ.
13. Socher R, Huang EH, Pennington J, Ng AY, Manning CD. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Adv Neural Inf Process Syst*. 2012;24:801-809.
14. Hermann KM, Blunsom P. The role of syntax in vector space models of compositional semantics. Paper presented at: Meeting of the Association for Computational Linguistics; 2013; Sofia, Bulgaria.
15. Lu Z, Li H. A deep architecture for matching short texts. Paper presented at: International Conference on Neural Information Processing Systems; 2013; Lake Tahoe, NV.
16. Erk K. A structured vector space model for word meaning in context. Paper presented at: Conference on Empirical Methods in Natural Language Processing; 2008; Honolulu, HI.
17. Lapata M, Mitchell J. Vector-based models of semantic composition. Paper presented at: Proceeding of Association for Computational Linguistics: Human Language Technologies; 2008; Columbus, OH.
18. Mitchell J, Lapata M. Composition in distributional models of semantics. *Cogn Sci*. 2010;34(8):1388.
19. Turney PD. Domain and function: a dual-space model of semantic relations and compositions. *J Artif Intell Res*. 2013;44(4):533-585.
20. Erk K. Vector space models of word meaning and phrase meaning: a survey. *Lang Linguist Compass*. 2012;6(10):635-653.
21. Clarke D. A context-theoretic framework for compositionality in distributional semantics. *Comput Linguist*. 2011;38(1):41-71.
22. Pennington J, Socher R, Manning C. GloVe: Global vectors for word representation. Paper presented at: Conference on Empirical Methods in Natural Language Processing; 2014; Doha, Qatar.
23. Das D, Smith NA. Paraphrase identification as probabilistic quasi-synchronous recognition. Paper presented at: Joint Conference of the Meeting of the ACL and the International Joint Conference on Natural Language Processing of the AFNLP: Volume 1; 2009; Suntec, Singapore.
24. Dolan B, Quirk C, Brockett C. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. Paper presented at: International Conference on Computational Linguistics; 2004; Geneva, Switzerland.
25. Nakov P, Zesch T. Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014): Association for Computational Linguistics and Dublin City University; 2014; Dublin, Ireland. http://www.aclweb.org/anthology/S14-2
26. Marelli M, Bentivogli L, Baroni M, Bernardi R, Menini S, Zamparelli R. Semeval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. Paper presented at: Semeval 2014: International Workshop on Semantic Evaluation; 2014; Dublin, Ireland.
27. Zhao J, Zhu T, Lan M. ECNU: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. Paper presented at: International Workshop on Semantic Evaluation; 2014; Dublin, Ireland.
28. Bjerva J, Bos J, van der Goot R, Nissim M. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. Paper presented at: Semeval-2014 Workshop; 2014; Dublin, Ireland.
29. Jimenez S, Dueñas GE, Baquero J, Gelbukh A. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. Paper presented at: Proceedings of the 8th International Workshop on Semantic Evaluation (SemeVal); 2014; Dublin, Ireland.
30. Lai A, Hockenmaier J. Illinois-LH: A denotational and distributional approach to semantics. Paper presented at: International Workshop on Semantic Evaluation; 2014; Dublin, Ireland.
31. Bestgen Y. CECL: A new baseline and a non-compositional approach for the sick benchmark. Paper presented at: International Workshop on Semantic Evaluation; 2014; Dublin, Ireland.
32. Proisl T, Evert S, Greiner P, Kabashi B. SemantiKLUE: Robust semantic similarity at multiple levels using maximum weight matching. Paper presented at: International Workshop on Semantic Evaluation; 2014; Dublin, Ireland.
33. Rus V, Mccarthy PM, Lintean MC, Mcnamara DS, Graesser AC. Paraphrase identification with lexico-syntactic graph subsumption. Paper presented at: International Florida Artificial Intelligence Research Society Conference, May 15-17, 2008; Coconut Grove, FL.
34. Blacoe W, Lapata M. A comparison of vector-based representations for semantic composition. Paper presented at: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning; 2012; Jeju Island, Korea.
35. Fernando S, Stevenson M. A Semantic Similarity Approach to Paraphrase Detection. Paper presented at: Computational Linguistics UK Annual Research Colloquium; 2009.

How to cite this article: Yao H, Liu H, Zhang P. A novel sentence similarity model with word embedding based on convolutional neural network. *Concurrency Computat Pract Exper*. 2018;e4415. https://doi.org/10.1002/cpe.4415