Neurocomputing 643 (2025) 130355

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Text similarity based on two independent channels: Siamese Convolutional Neural Networks and Siamese Recurrent Neural Networks

Zhengfang He

School of Intelligent Science and Engineering, Yunnan Technology and Business University, Kunming, 650000, Yunnan, China

ARTICLE INFO

Communicated by E. Verdu

Keywords:
Text similarity
Siamese Convolutional Neural Networks
Siamese Recurrent Neural Networks

ABSTRACT

In the present-day context, a large amount of information exists in text. It is hard to extract meaningful and potential information from the text. From the current research, text similarity provides a method applied in many practical scenarios. Traditional text similarity algorithms are easy to implement, but more than these algorithms are needed to extract text features. At present, most text similarity algorithms are based on deep learning. However, these algorithms often struggle with adequately extracting both local and context text features, and they typically do not differentiate between the effectiveness of these two types of feature extraction. To address these shortcomings, this paper proposes Two Independent Channels: Siamese Convolutional Neural Networks and Siamese Recurrent Neural Networks (TIC-SCNN-SRNN). This approach is designed for binary classification, where '1' indicates similarity and '0' indicates dissimilarity. In detail, this paper proposes the Siamese Convolutional Neural Networks (SCNN) model to address the issue of insufficient extraction of local text features. Additionally, it introduces the Siamese Recurrent Neural Networks (SRNN) model to tackle the problem of insufficient extraction of context text features. Due to the issue of not distinguishing the effects of local and context text features extractions, this paper conducts independent weight learning on these two models to research which is more effective for text similarity tasks. In order to verify the effectiveness of the model, this paper experiments on SciTail, TwitterPPD, and QQP datasets. The experimental results show that SCNN and SRNN influence the text similarity tasks, but SRNN is more effective than SCNN. Furthermore, to verify the advantages of the TIC-SCNN-SRNN model, this paper tests the performance of the state-of-the-art CNN&RNN models. The test results show that the TIC-SCNN-SRNN model performs the best, indicating that the model proposed in this paper is more effective for the text similarity tasks.

1. Introduction

With the rapid development of science and technology, people's living levels continue to improve. Internet technology has undergone unprecedented development and has produced massive amounts of information. All kinds of information put people at a loss when searching for the information they need. People hope to obtain the content they need quickly from the Internet. Quickly finding helpful information is an urgent problem to be solved in the Internet development [1,2].

In order to retrieve relevant information from massive data quickly, a core technology needs to be applied. That is text similarity. This technique filters out invalid information and is widely used in various scenarios [3]. In database redundancy detection, if the similarity between two text paragraphs is extremely high, the redundant data can be deleted to optimize the database storage [4]. In information retrieval, this technique filters the search results to identify those most similar to the query text [5,6]. In the automatic question-answering system, the text similarity detects the matching level between the user's question and the answers in the corpus to improve the response accuracy [7,8]. In text clustering, the related documents are clustered according to the similarity of the text [9]. Text similarity serves as fundamental research that connects to upper-level applications. By advancing the study of text similarity, the performance of these applications can be significantly enhanced.

Lin explained that text similarity was related to the differences between texts from information theory [10]. The larger the difference, the lower the similarity; conversely, the smaller the difference, the higher the similarity, as illustrated in Eq. (1).

$$sim(A,B) = \frac{I(common(A,B))}{I(description(A,B))} \tag{1}$$

where *common(A,B)* is a proposition that states the commonalities between text *A* and text *B*; *description(A,B)* is a proposition that describes what text *A* and text *B* are. *I(s)* is the information contained in a proposition *s*. In information theory, the information content of a

E-mail address: hfrommane@qq.com.

https://doi.org/10.1016/j.neucom.2025.130355
Received 18 July 2023; Received in revised form 26 February 2025; Accepted 26 April 2025
Available online 12 May 2025

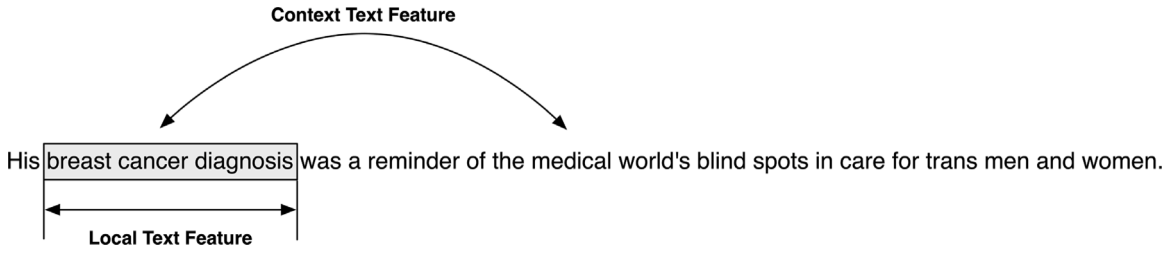

Fig. 1. Schematics of local text feature and context text feature.

statement is measured by the negative logarithm of the probability of that statement. Therefore,

$$I\ (common(A,B)) = -logP(common(A,B))$$
$$I\ (description(A,B)) = -logP(description(A,B))$$

Further, the similarity between text A and text B is measured by the ratio between the amount of information needed to state the commonalities of text A and text B, the information needed to fully describe what text A and text B are, as shown in Eq. (2).

$$sim(A,B) = \frac{logP(common(A,B))}{logP(description(A,B))} \tag{2}$$

The text similarity can generally be represented by a real number in the range (0,1], and this numerical value can be calculated based on the semantic distance between the texts.

Extraction of text features can be divided into local text features and context text features. As shown in Fig. 1, the "*breast cancer diagnosis*" is a local text feature, and the relationship between "*cancer*" and "*medical*" is defined as a context text feature. In general, features comprising 3–5 consecutive words are defined as local text features, while relationships spanning more than 5 words are defined as context text features.

Text similarity algorithms include traditional algorithms (the specific algorithms described in Section 2.1) and deep learning algorithms (the specific algorithms described in Section 2.2). The traditional algorithms are easy to implement; however, more advanced methods are required to extract context text features effectively. Algorithms based on deep learning include Deep Neural Network (DNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and integrating Convolutional Neural Network & Recurrent Neural Network (CNN&RNN) [11].

Many researchers have used CNN to perform text similarity tasks [12,13], but due to the inherent characteristics of CNN, CNN-based algorithms are more suitable for extracting local text features. Also, many researchers have utilized RNN for text similarity tasks [14–17]. RNN is particularly suitable for extracting context text features. Furthermore, some researchers have integrated CNN and RNN for text similarity tasks [18–23], yet these models need to distinguish between the effectiveness of local and context text features. In summary, current text similarity algorithms based on deep learning often struggle with adequately extracting both local and context text features, and they typically do not differentiate between the effectiveness of these two types of feature extraction.

Therefore, this paper proposes Two Independent Channels: Siamese Convolutional Neural Networks and Siamese Recurrent Neural Networks (TIC-SCNN-SRNN), to address these shortcomings. This approach is designed for binary classification, where '1' indicates similarity and '0' indicates dissimilarity. In detail, this paper **proposes the Siamese Convolutional Neural Networks (SCNN) model to address the issue of insufficient extraction of local text features**. Additionally, it **introduces the Siamese Recurrent Neural Networks (SRNN) model to tackle the problem of insufficient extraction of context text features**. Because of the problem of not distinguishing the effects of local and context text features extractions, this paper **conducts independent weight learning on these two models to research which is more effective for text similarity tasks**.

This paper is structured as follows: Section 2 provides a comprehensive literature review. Section 3 discusses the methodology. Section 4 introduces the paper's core, detailing the Two Independent Channels: Siamese Convolutional Neural Networks and Siamese Recurrent Neural Networks. Section 5 presents the experiments and analyses. Finally, Section 6 concludes the paper, summarizing the findings and suggesting potential directions for future research.

2. Literature review

2.1. Traditional text similarity algorithms

Text similarity calculation is fundamental research in natural language processing and has a long history. Traditional algorithms for calculating text similarity can be broadly categorized into three types: string-based, corpus-based, and knowledge-based [24].

2.1.1. String-based methods

String-based methods involve a direct comparison of the original text, mainly including Levenshtein Distance (LD) [25], Longest Common Sequence (LCS) [26], N-Gram [27], and Jaccard Similarity [28]. These methods are simple in principle and easy to implement, making them suitable for quick fuzzy text matching. However, their drawbacks lie in the inability to analyze word meanings, relationships between words, and the incapacity to handle synonyms.

2.1.2. Corpus-based methods

The corpus-based algorithms were derived from the distributional hypothesis proposed by Harris in 1954 [29]. This hypothesis is that words with similar contexts should have similar semantics. Its calculation is utterly dependent on the corpus, and its semantic similarity is measured according to the co-occurrence frequency of words in the text. Currently, the primary methods for calculating text similarity based on a corpus involve representing texts as vectors. Different ways of constructing vectors include the Vector Space Model (VSM) [30], Latent Semantic Analysis (LSA) [31], Probabilistic Latent Semantic Analysis (PLSA) [32], and Latent Dirichlet Allocation (LDA) [33].

2.1.3. Knowledge-based methods

The knowledge-based methods utilize a knowledge base with a standardized organizational system to calculate text similarity, generally divided into ontology-based and network-based knowledge. Algorithms based on ontology knowledge use the interrelationships between concepts in the ontology structure system to calculate text similarity, such as WordNet [34]. In the algorithms based on network knowledge, the entries are structured, and hyperlinks connect the entries. Mainly including WikiRelate! [35], Explicit Semantic Analysis (ESA) [36].

2.2. Deep learning text similarity algorithms

2.2.1. Early deep learning-based model

Huang et al. proposed the Deep Structured Semantic Models (DSSM) algorithm [37]. This model is one of the first algorithms to use the Siamese Networks architecture for semantic text similarity calculation. The DSSM architecture is divided into an input layer, a presentation layer, and a matching layer. The input layer primarily maps the original text to a vector. The presentation layer maps a high-dimensional sentence vector to a low-dimensional vector through several fully connected layers. The matching layer calculates the cosine similarity of two low-dimensional vectors to characterize the semantic similarity of two sentences. Although the DSSM model has demonstrated outstanding results in text-matching tasks, it overlooks the local and context features of the text.

2.2.2. CNN-based models

In the CNN-based text similarity calculation research, Severyn and Moschitti proposed the Convolutional Deep Neural Network (CDNN) model for short text rearrangement [12]. This model utilizes the Query-Document structure. Consequently, the two Neural Networks operate independently, with no weight sharing. This architecture proves to be more suitable for Question Answering systems. He et al. introduced multi-perspective sentence similarity modeling with Convolutional Neural Networks [13]. The model features Convolutional Neural Networks with multiple granularity levels and various pooling types. It utilizes the structure of Siamese Convolutional Neural Networks, where the two Convolutional Neural Networks share weights. Nevertheless, this structure does not consider the context features of the text.

2.2.3. RNN-based models

In the research of text similarity calculation based on RNN, Mueller and Thyagarajan proposed the Manhattan LSTM (MaLSTM) model of sentence similarity [14]; Neculoiu et al. presented the character-based Siamese Bidirectional Long Short-Term Memory (BiLSTM) model [15]; de Souza et al. introduced the Siamese Neural Networks on short text similarity tasks for multiple domains and languages [16]; Lv et al. proposed Siamese Multiplicative LSTM for semantic text similarity [17]. All these models utilize a Siamese architecture and LSTM to extract text features. However, this paper believes these models need more extraction of local text features.

2.2.4. Hybrid CNN-RNN models

Indeed, some researchers have integrated CNN and RNN to analyze texts. Linhares Pontes et al. proposed the Siamese CNN and LSTM model [18]. This model uses the output of CNN as part of RNN's input. The main contribution of this model comes from LSTM, so the extraction of text features needs to be improved. Many researchers have employed sequential methods to integrate CNN and RNN. Among them, Zhou [21] and Balita et al. [22] used the output of CNN as the input of RNN, while Zhang et al. [23] and Niu et al. [20] used the opposite method to integrate.

Zhou used Term Frequency-Inverse Document Frequency (TF-IDF) to eliminate features with lower weights, extracted key features from the text, and then inputted them into the CNN-LSTM model [21]. While this approach can achieve training acceleration and parameter reduction, it leads to a loss of accuracy caused by the TF-IDF feature extraction method. Balita et al. employed a sentiment analysis model using CNN and LSTM frameworks to analyze sentiment in book reviews [22]. The hybrid models are able to maximize efficiency due to the integration of both algorithms. However, in these two architectures, the CNN and LSTM are sequential (the input of LSTM is the output of CNN), and the LSTM does not process the original texts, which results in some loss of information from the original texts. Hence, more than the extraction of text features is required.

Zhang et al. presented a text classification method based on the LSTM-CNN model, which combined the benefits of LSTM and CNN [23]. The model extracts semantic information from text sentences using the LSTM model. Subsequently, it employs the CNN model to extract local features. The experimental findings reveal that the LSTM-CNN hybrid model effectively improves the accuracy of text classification. Niu et al. proposed a feature expansion and Siamese Network model [20]. First, the model uses Latent Dirichlet Allocation (LDA) to expand the features of short texts, then uses a Convolutional Neural Network (CNN) and a Bi-directional Long Short-Term Memory (BiLSTM) to extract deep features. This model employs the Siamese architecture to calculate text similarity. Experimental results indicate that, based on the dataset from the Ant Financial NLP Challenge, this method achieves higher accuracy and F1 score. Likewise, these two architectures are also sequential models. The difference is that the input of CNN is the output of LSTM. Since CNN does not directly process texts, they also lead to losses of text features.

Ameur et al. proposed Stacked-RNN-CNN and Combined-RNN-CNN [19]; these models integrated RNN and CNN for Arabic text categorization. The experimental results on the Arabic dataset demonstrate high accuracy. However, these models do not use the structure of Siamese Neural Networks; they are more suitable for text classification tasks. The RNN and CNN in these models are not independent, and there is no distinction between context and local text features.

2.2.5. Transformer-based models

In recent years, significant progress has been made in text similarity methods with the emergence of transformer-based models. Models such as BERT [38], ALBERT [39], and DeBERTa [40] have been successfully adapted for text similarity tasks by fine-tuning on sentence-pair datasets. While these methods excel in capturing global contextual representations, they often incur high computational costs and may not fully capture fine-grained local text features.

Furthermore, some recent studies have attempted to integrate transformer models with traditional CNN [41] or RNN [42] architectures to balance the extraction of local or context text features. These hybrid models indicate promising directions for addressing existing shortcomings.

2.3. Summary

This review highlights that traditional text similarity methods, while simple and efficient, are limited by their inability to capture semantic nuances beyond surface-level features. Deep learning approaches such as DSSM, CNN-based, and RNN-based models have substantially improved semantic representation. However, many of these methods either focus predominantly on either local or contextual features or adopt a sequential integration that can lead to information loss. Even hybrid models that combine CNN and RNN typically lack a clear, independent treatment of these two feature types.

Moreover, while recent transformer-based models have achieved impressive performance, they tend to emphasize global contextual information at the expense of local details and involve considerable computational overhead.

In light of these limitations, our proposed approach, TIC-SCNN-SRNN, offers a novel solution. By independently learning local and context text features and then integrating them through separate channels, our method effectively leverages the strengths of both CNN and RNN architectures, providing enhanced performance in text similarity tasks.

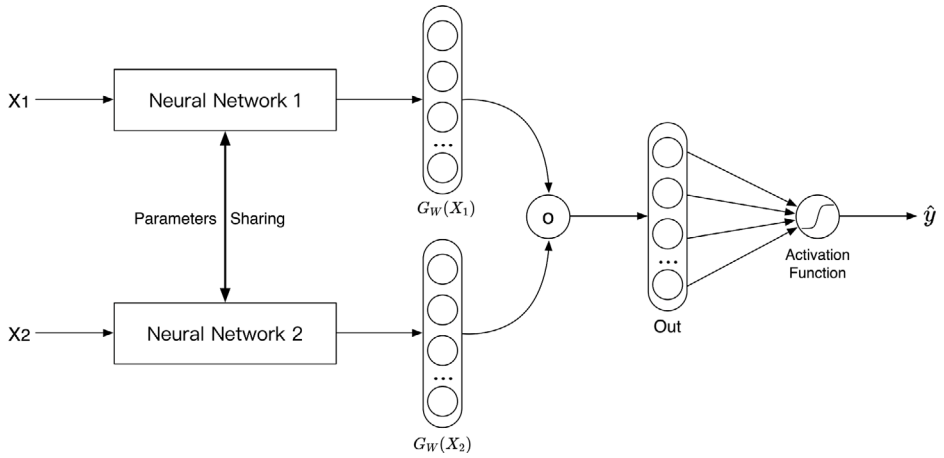

Fig. 2. The architecture of the Siamese Neural Networks.

3. Methodology

3.1. The basic architecture of the Siamese neural networks

For deep learning, the training process usually requires many samples. In contrast, humans can learn to distinguish objects from a single sample. For instance, show children a picture of an elephant; they can recognize which animal is the elephant when they go to the zoo. Similarly, deep learning algorithms also aim to converge the models with minimal samples, acknowledging the challenges of collecting and labeling a large amount of data. This challenge is commonly referred to as the One-shot Learning problem.

An easy way to solve this problem is transfer learning. Start by training a SoftMax classifier on a massive dataset, then use the One-shot Learning dataset to fine-tune the parameters of the last layer. Due to the limited number of samples in each category in the One-shot Learning dataset, even if the parameters of the last layer are fine-tuned, the model may still overfit. Solving this problem with a similar method is challenging, so the Siamese Neural Networks is proposed, as shown in Fig. 2.

The concept of the Siamese Neural Networks can be summarized as follows: Simultaneously feed the Neural Networks with two samples and task them with predicting whether these samples belong to the same category. In the testing phase, the test sample to be identified is compared with each sample in the training set, and the output indicates which sample in the training set belongs to the same category as the test sample. In simpler terms, two samples are input into the Neural Networks simultaneously, and the output is the probability that they belong to the same category [43].

If x_1 and x_2 represent two samples in the dataset, let $x_1 \circ x_2$ indicate that x_1 and x_2 belong to the same category. Note that $x_1 \circ x_2$ and $x_2 \circ x_1$ are equivalent, which means that the output probability remains the same if the order of the input samples is reversed, as shown in Eq. (3).

$$P\left(x_1 \circ x_2\right) = P\left(x_2 \circ x_1\right) \tag{3}$$

This is the symmetry of the Siamese Neural Networks, which strongly relies on this design. In order to make the Neural Networks symmetrical, the parameters of the two networks must be shared. Then, the difference between the outputs of the two networks is fed into a classifier.

In Fig. 2, the "Siamese" of the Siamese Neural Networks is realized through parameters sharing. In other words, the Siamese Neural Networks inputs X_1 and X_2 into *Network 1* and *Network 2*, respectively. These two networks map the input to the target space as $G_W\left(X_1\right)$ and $G_W\left(X_2\right)$, then calculate the distance within this space. This distance is the "semantic" representation of the input [44].

3.2. Siamese neural networks for text similarity task

The basic structure of the text similarity model based on Siamese Neural Networks is illustrated in Fig. 2. X_1 and X_2 are a pair of input texts, and *Neural Network 1* and *Neural Network 2* are a pair of Neural Networks. The parameters of these two Neural Networks are shared.

In Fig. 2, $G_W\left(X_1\right)$ and $G_W\left(X_2\right)$ are two low-dimensional vectors mapped through the Neural Networks. 'Out' describes the difference between these two vectors, calculated as the square of their subtraction; this is expressed by Eq. (4) [44].

$$Out = \left(G_W\left(X_1\right) - G_W\left(X_2\right)\right)^2 \tag{4}$$

Activation Function is the Sigmoid function [45]; the $\hat{y}$ is the model's output, a value between (0,1).

The model uses the Sigmoid function as the activation function for the last layer, transforming the entire model into a logistic regression model. Logistic regression is a supervised learning model that solves binary classification problems. It aims to minimize the difference between the predicted value $\hat{y}$ and the true value y.

4. Two independent channels: Siamese convolutional neural networks and Siamese recurrent neural networks

4.1. Overall model

The existing text similarity algorithms based on deep learning suffer from issues such as inadequate extraction of both local and context text features and no distinction between the effectiveness of these two types of feature extraction. In response to these shortcomings, this paper proposes Two Independent Channels: Siamese Convolutional Neural Networks and Siamese Recurrent Neural Networks (TIC-SCNN-SRNN).

The TIC-SCNN-SRNN model, as shown in Fig. 3, consists of Siamese Convolutional Neural Networks (SCNN) and Siamese Recurrent Neural Networks (SRNN), with the two models operating independently. The input to the model is a pair of texts, $T_1 \ldots T_n$ and $T_1' \ldots T_n'$, and it is fed into SCNN and SRNN, respectively. $\hat{y}_{SCNN}$ and $\hat{y}_{SRNN}$ are the outputs of SCNN and SRNN, respectively. After getting $\hat{y}_{SCNN}$ and $\hat{y}_{SRNN}$, the losses of SCNN and SRNN are calculated separately according to the logistic regression algorithm. Subsequently, *Loss1* and *Loss2* are scaled to get the final *Loss*. Finally, gradient descent and backpropagation are used to train the TIC-SCNN-SRNN model. The construction of the TIC-SCNN-SRNN model encompasses the following 5 steps:

STEP 1:

In general, the TIC-SCNN-SRNN model trains two Embeddings, firstly. *Embeddings_C* for SCNN and *Embeddings_R* for SRNN need to be jointly trained with TIC-SCNN-SRNN. Furthermore, the outputs of

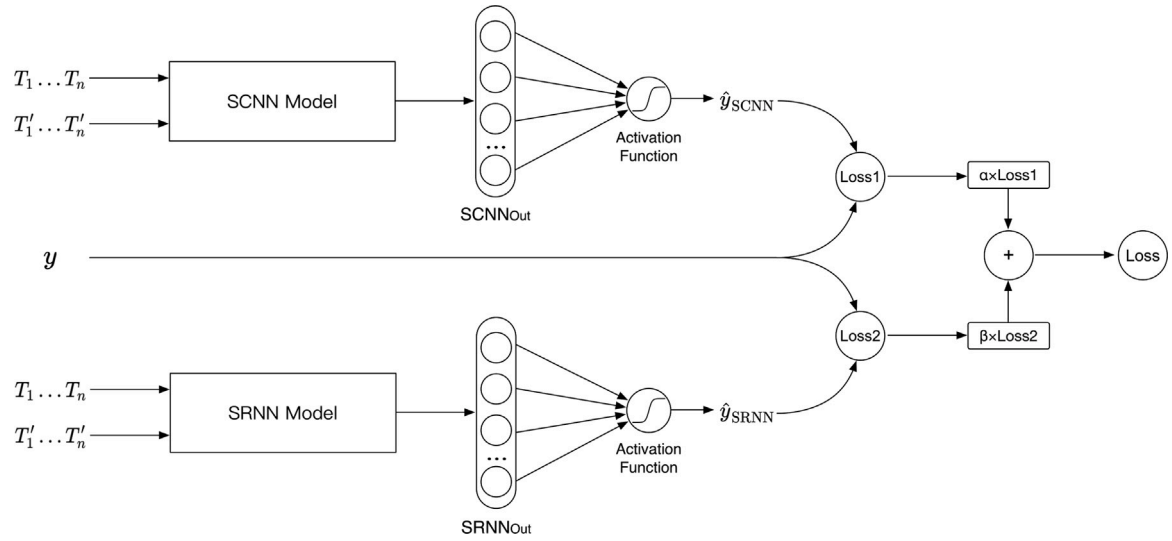

Fig. 3. The TIC-SCNN-SRNN model.

Table 1
Data examples of the overall model.

ID	$T_1 \dots T_n$	$T'_1 \dots T'_n$	Y
1	I like the city of Kunming.	I love the charming city of Kunming.	1
2	I like hamburgers.	Taylor Swift's music is good.	0

Table 2
The combination of α and β.

Name	α (for SCNN)	β (for SRNN)
Model_1	1.0	0.0
Model_2	1.0	0.000001
Model_3	0.0	1.0
Model_4	0.000001	1.0
Model_5	1.0	1.0

Embeddings_C and *Embeddings_R* are used as the inputs of the SCNN and the SRNN, respectively.

STEP 2:

Following the extraction of text features by the SCNN and the SRNN, two low-dimensional vectors are obtained, namely $SCNN_{out}$ (the output of SCNN) and $SRNN_{out}$ (the output of SRNN).

STEP 3:

Then, each of the two low-dimensional vectors is passed through a fully connected neural network followed by a Sigmoid activation function to obtain the outputs of the two models, namely $\hat{y}_{SCNN}$ for SCNN and $\hat{y}_{SRNN}$ for SRNN.

STEP 4:

Furthermore, the logistic regression algorithm is used to calculate the losses for SCNN and SRNN, respectively. SCNN's loss is *Loss1*, and SRNN's loss is *Loss2*.

STEP 5:

Finally, scale *Loss1* and *Loss2* appropriately, multiplying them by α and β respectively, and then sum the $\alpha \times Loss1$ and $\beta \times Loss2$ to calculate the final *Loss*.

In Fig. 3, $T_1 \dots T_n$ and $T'_1 \dots T'_n$ are a pair of input texts; Y is the label for these texts, with values 0 or 1, where 0 indicates that the texts are dissimilar, and 1 indicates that the texts are similar. The model's output is a probability value between 0 and 1; the closer it is to 1, the greater the probability that the texts are similar. Conversely, the closer it is to 0, the greater the probability that the texts are dissimilar. A specific input example of this model is shown in Table 1. The first data, "*I like the city of Kunming.*" is represented by $T_1 \dots T_n$, and "*I love the charming city of Kunming.*" is represented by $T'_1 \dots T'_n$. The pair of texts is similar, so Y is 1. The second data, "*I like hamburgers.*" and "*Taylor Swift's music is good.*", is dissimilar, so Y is 0.

It should be noted that the values of α and β in this paper can be 0, 0.000001, or 1. These configurations allow for researching which of SCNN and SRNN is more effective for text similarity through different combinations of α and β. The combinations of α and β are shown in Table 2.

Model_1 set $\alpha = 1.0$ and $\beta = 0.0$, indicating that only SCNN is working; it aims to research the effectiveness of SCNN on text similarity tasks.

Model_2 set $\alpha = 1.0$ and $\beta = 0.000001$, meaning both SCNN and SRNN are working. By comparing the experimental results of Model_1 and Model_2, it can research the effectiveness of SRNN for text similarity tasks.

Model_3 set $\alpha = 0.0$ and $\beta = 1.0$, indicating that only SRNN is working; it aims to research the effectiveness of SRNN on text similarity tasks.

Model_4 set $\alpha = 0.000001$ and $\beta = 1.0$, meaning both SCNN and SRNN are working. By comparing the experimental results of Model_3 and Model_4, it can research the effectiveness of SCNN for text similarity tasks.

Model_5 set $\alpha = 1.0$ and $\beta = 1.0$, giving equal weight to both SCNN and SRNN in the TIC-SCNN-SRNN model. By comparing the experimental results of Model_2, Model_4, and Model_5, it can research which of SCNN and SRNN is more effective for text similarity tasks.

4.2. Embedding layer

The embedding layer converts each word into a fixed-length vector of a defined size. In this paper, a fully connected Neural Network is used as the Embedding layer.

The initial weight of the Embedding output matrix in this paper is not randomly initialized. Instead, a large matrix is employed by utilizing the word2vec [46] pre-training model on all dictionary words. Since the TIC-SCNN-SRNN model comprises two independent channels, two Embeddings are trained for SCNN and SRNN, named *Embeddings_C* and *Embeddings_R*. Notably, in this model, the Embeddings are not trained independently but are jointly trained with the TIC-SCNN-SRNN model.

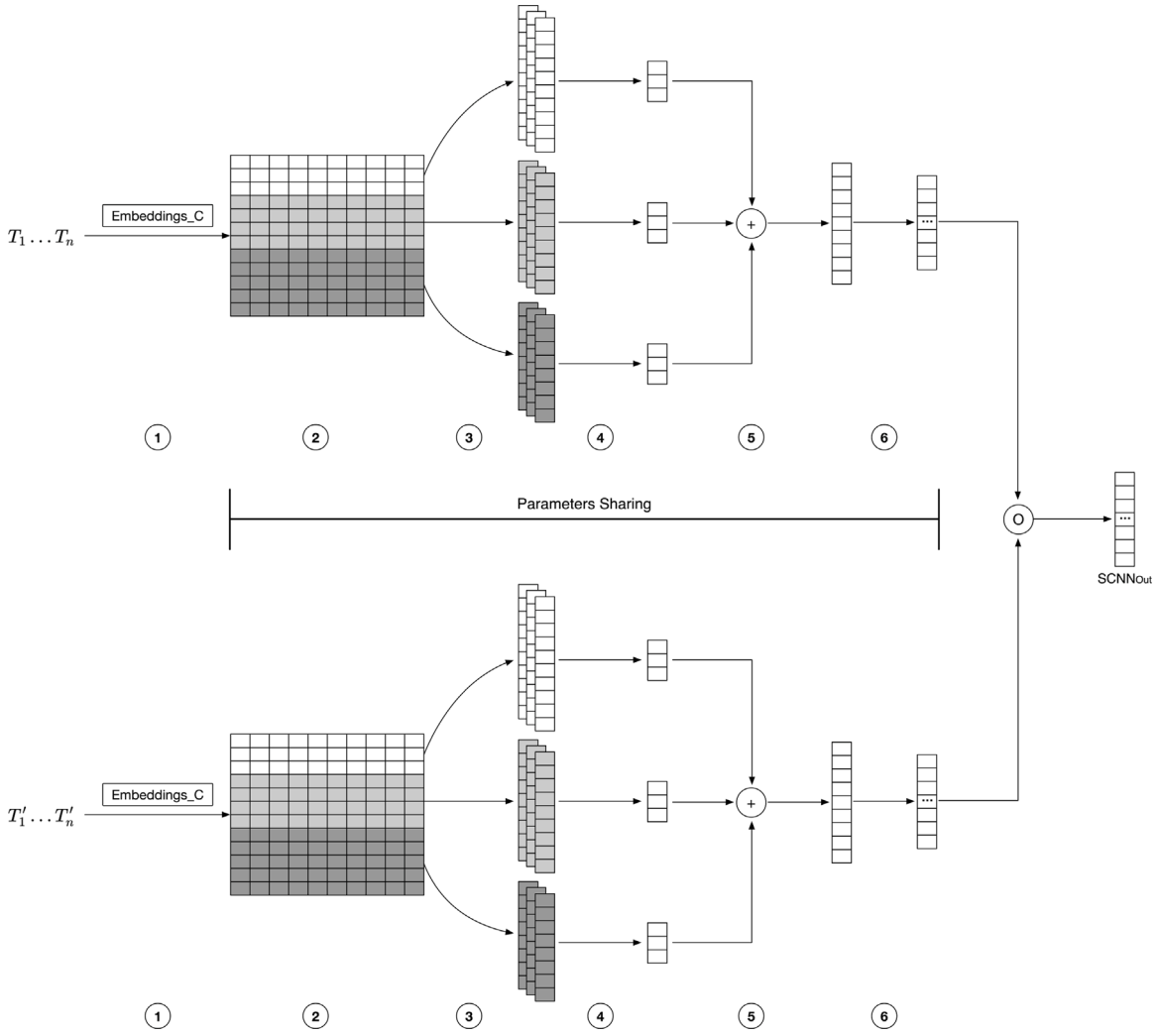

Fig. 4. The SCNN model.

4.3. Siamese convolutional neural networks

In order to solve the problem of insufficient extraction of local text features, this paper proposes the SCNN model, leveraging the TextCNN architecture for enhanced text processing effectiveness. The TextCNN, proposed by Kim in 2014 [47], embodies the concept of N-grams. SCNN is based on TextCNN and uses three sizes of convolution kernels to extract essential information from sentences, thereby improving the capture of local features. There are 6 steps in the model, as shown in Fig. 4.

STEP 1&2:

In TextCNN, a sentence is viewed as a sequence of words of length n. Each word vector is represented by x_i, and the Embeddings dimension of each word is k. The expression of the sentence is shown in Eq. (5). The weights of Embeddings in this paper are initialized by word2vec, and then the weights of Embeddings are updated with training.

$$x_{1:n} = x_1 \oplus x_2 \oplus \cdots \oplus x_n \tag{5}$$

STEP 3:

The convolution of TextCNN is one-dimensional, the width of the convolution kernel matches the dimension of Embeddings, and the height of the convolution kernel h represents the number of words that the sliding window covers in each slide. This paper uses three convolution kernels with different heights, namely h_1, h_2, and h_3; each convolution kernel is $w \in \mathbb{R}^{hk}$. The result of each convolution operation via the sliding window is c_i, as shown in Eq. (6). Moreover, in the equation, $b \in \mathbb{R}$ and the function f is non-linear (such as tanh or ReLU).

$$c_i = f\left(w \cdot x_{i:i+h-1} + b\right) \tag{6}$$

The sequence length of the input sentence is n, and the height of the convolution kernel is h. Consequently, the convolution kernel slides $n-h+1$ (including the start position) times in total, and the output result of the convolution after sliding is shown in Eq. (7).

$$c = \left[c_1, c_2, \dots, c_{n-h+1}\right] \tag{7}$$

STEP 4:

The pooling of TextCNN uses Max-pooling, that is, $\hat{c} = max\{c\}$. Typically, there are multiple convolution kernels for each height, which is m in this paper. The pooled data is shown in Eq. (8), and its dimension is $1 \times m$. Due to there are three types of convolution kernels with different heights, three z will be generated by Eq. (8); they are z_1, z_2, and z_3.

$$z = \left[\hat{c}_1, \hat{c}_2, \dots, \hat{c}_m\right] \tag{8}$$

STEP 5:

After pooling, the vectors z_1, z_2, and z_3 should be concatenated as described in Eq. (9).

$$\acute{z} = z_1 \oplus z_2 \oplus z_3 \tag{9}$$

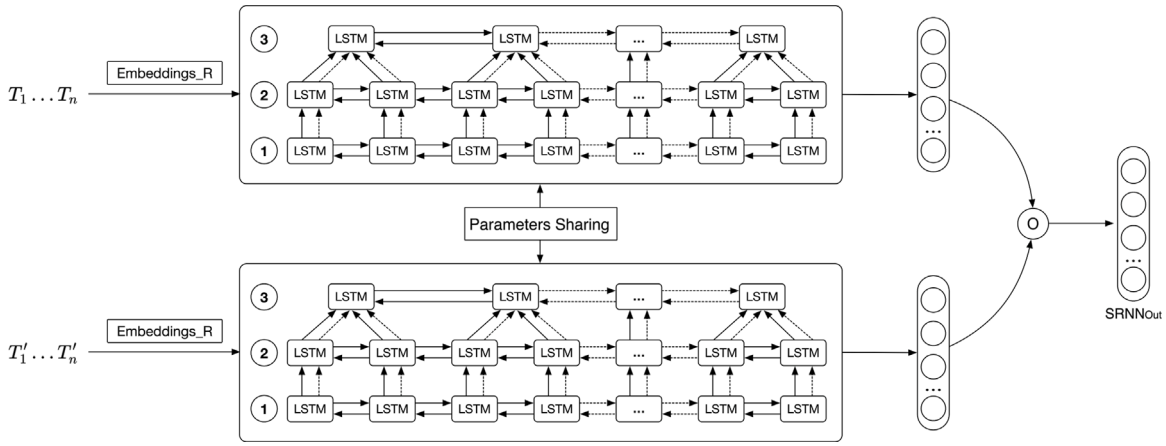

Fig. 5. The SRNN model.

STEP 6:

The dimensionality of the vector obtained after Max-pooling is higher. Hence, this paper uses a fully connected Neural Network layer to reduce the vector to 128 dimensions, as shown in Eq. (10).

$$CNN_{out} = g\left(W \cdot \acute{z} + b\right) \tag{10}$$

This paper uses the Siamese structure; the output of SCNN is shown in Eq. (11).

$$SCNN_{out} = \left(CNN_{out T_1 \cdots T_n} - CNN_{out T'_1 \cdots T'_n}\right)^2 \tag{11}$$

4.4. Siamese recurrent neural networks

In order to solve the problem of insufficient extraction of context text features, this paper proposes the Siamese Recurrent Neural Networks (SRNN) model. As shown in Fig. 5, the SRNN has two inputs, $T_1 \dots T_n$ and $T'_1 \dots T'_n$, respectively. The SRNN model uses three Bidirectional LSTM [48] layers.

LAYER 1:

After embedding, the first layer of RNN is shown in Eq. (12). At each time-step t, this layer maintains two hidden layers, one for the left-to-right propagation and another for the right-to-left propagation.

$$\begin{aligned} \overrightarrow{h}_t^1 &= f\left(\overrightarrow{W}^1 x_t + \overrightarrow{V}^1 \overrightarrow{h}_{t-1}^1 + \overrightarrow{b}^1\right), \text{ where } t \in [1, n] \\ \overleftarrow{h}_t^1 &= f\left(\overleftarrow{W}^1 x_t + \overleftarrow{V}^1 \overleftarrow{h}_{t+1}^1 + \overleftarrow{b}^1\right), \text{ where } t \in [1, n] \end{aligned} \tag{12}$$

LAYER 2:

The second layer of this model is shown in Eq. (13), where the input (the part from LAYER 1) to each intermediate neuron is the output of the RNN at LAYER 1 at the same time-step t. Moreover, in this layer, at time-step t, each intermediate neuron receives one set of parameters from the previous time-step of LAYER 2 and two sets of parameters from LAYER 1 (one is the left-to-right RNN, and the another is the right-to-left RNN).

$$\begin{aligned} \overrightarrow{h}_t^2 &= f\left(\overrightarrow{W}^2 h_t^1 + \overrightarrow{V}^2 \overrightarrow{h}_{t-1}^2 + \overrightarrow{b}^2\right), \text{ where } t \in [1, n] \\ \overleftarrow{h}_t^2 &= f\left(\overleftarrow{W}^2 h_t^1 + \overleftarrow{V}^2 \overleftarrow{h}_{t+1}^2 + \overleftarrow{b}^2\right), \text{ where } t \in [1, n] \end{aligned} \tag{13}$$

LAYER 3:

The third layer of this model is shown in Eq. (14), where the input (the part from LAYER 2) of this layer at time t is the average of the outputs of LAYER 2 at time $2t-1$ and $2t$. Consequently, the number of neurons in this layer is reduced by half.

$$\begin{aligned} \overrightarrow{h}_t^3 &= f\left(\overrightarrow{W}^3 \frac{h_{2t-1}^2 + h_{2t}^2}{2} + \overrightarrow{V}^3 \overrightarrow{h}_{t-1}^3 + \overrightarrow{b}^3\right), \text{ where } t \in \left[1, \frac{n}{2}\right] \\ \overleftarrow{h}_t^3 &= f\left(\overleftarrow{W}^3 \frac{h_{2t-1}^2 + h_{2t}^2}{2} + \overleftarrow{V}^3 \overleftarrow{h}_{t+1}^3 + \overleftarrow{b}^3\right), \text{ where } t \in \left[1, \frac{n}{2}\right] \end{aligned} \tag{14}$$

After processing through these three layers, the output of RNN can be obtained, as shown in Eq. (15).

$$RNN_{\text{out}} = g\left(U\left[\overrightarrow{h}_t^3; \overleftarrow{h}_t^3\right] + c\right) \tag{15}$$

Thus, the output of SRNN is shown in Eq. (16).

$$SRNN_{out} = \left(RNN_{out T_1 \cdots T_n} - RNN_{out T'_1 \cdots T'_n}\right)^2 \tag{16}$$

RNN neurons are generally used in Siamese Recurrent Neural Networks. Standard RNN has feedback loops in the recursive layer, allowing it to maintain memory information over time. However, training standard RNN for tasks requiring long-term dependency is challenging due to the exponential decay of the gradient of the loss function during training, leading to the vanishing gradient problem.

Long Short-Term Memory (LSTM) [49] is a powerful type of RNN used in deep learning. LSTM can effectively prevent the problem of vanishing gradient, which is the main limitation of RNN. It has a memory that can maintain its state over time, and internal mechanisms called gates can regulate the flow of information. Given these advantages, this paper employs LSTM as the neuron for SRNN.

4.5. Model loss

SCNN and SRNN can be regarded as logistic regression models; their predicted values and loss functions are shown in Eqs. (17) and (18). The loss function of the TIC-SCNN-SRNN model is shown in Eq. (19), where *Loss1* and *Loss2* are scaled appropriately, multiplied by α and β, respectively, and then summated the $\alpha \times Loss1$ and $\beta \times Loss2$ to get the final *Loss*. Subsequently, training the model with gradient descent and backpropagation.

$$\begin{aligned} \hat{y}_{SCNN} &= Sigmoid\left(W \cdot SCNN_{out} + b\right) \\ \hat{y}_{SRNN} &= Sigmoid\left(W \cdot SRNN_{out} + b\right) \end{aligned} \tag{17}$$

$$\begin{aligned} \text{Loss}_1 &= L\left(\hat{y}_{SCNN}, y\right) \\ &= -\left(y \log \hat{y}_{SCNN} + (1-y) \log\left(1 - \hat{y}_{SCNN}\right)\right) \\ \text{Loss}_2 &= L\left(\hat{y}_{SRNN}, y\right) \\ &= -\left(y \log \hat{y}_{SRNN} + (1-y) \log\left(1 - \hat{y}_{SRNN}\right)\right) \end{aligned} \tag{18}$$

$$Loss = \alpha \times Loss_1 + \beta \times Loss_2 \tag{19}$$

5. Experiments and analyses

5.1. Experimental hardware and software

Many studies have shown that GPU is much faster than CPU for deep learning algorithms [50,51], so this paper uses GeForce RTX 2080 as

Table 3
The primary hardware models.

Hardware	Model
CPU	Intel Core i5-9400F @ 2.90 GHz
RAM	16 GB (DDR4 3200 MHz)
GPU	Nvidia GeForce RTX 2080 SUPER (8 GB)

Table 4
The main software versions.

Software	Version
Windows	10
CUDA	9.0
cuDNN	7.6.3
Python	3.5.2
TensorFlow-GPU	1.11.0
Keras	2.2.4

Table 5
The basic information of the datasets.

Dataset	Label	Amount
SciTail	Two	27 k
TwitterPPDB	Two	51.5 k
QQP	Two	400 k

Table 6
Datasets statistics.

Dataset	Max	Min	Mean	Median
SciTail	103	1	14.5	13.0
TwitterPPDB	102	1	14.8	14.0
QQP	237	1	11.1	10.0

the GPU for the experiments. The primary hardware models are shown in Table 3.

This paper uses GPU to accelerate the training of Neural Network. Hence, cuDNN (a GPU-accelerated library) needs to be installed, and CUDA (a parallel computing framework) needs to be installed before installing cuDNN. At the same time, this paper selects TensorFlow [52] and Keras (an open-source high-level Neural Network library written in Python and running on TensorFlow) as the deep learning frameworks. The main software versions are shown in Table 4.

5.2. Datasets and data pre-processing

High-quality public datasets are crucial for research in a field; this paper chooses SciTail, TwitterPPDB, and QQP datasets for the experiments, as shown in Table 5.

The SciTail dataset is an entailment dataset created from multiple-choice science exams and web sentences [53]. This dataset annotates the sentence pair as supports (entails) or not (neutral). The dataset contains 27,026 examples, with 10,101 having entail labels and 16,925 having neutral labels.

The TwitterPPDB dataset collects sentence-level paraphrases from Twitter by linking tweets through shared URLs. This corpus has 51,524 human-annotated sentence pairs, each labeled with 0 or 1 [54].

Quora Question Pairs (QQP) dataset consists of over 400,000 question pairs from the Quora website, and each question pair is annotated with a binary value indicating whether the two questions are paraphrased of each other [55].

Data preprocessing has a significant influence on the accuracy of the Neural Network, so this paper uses the following two methods to preprocess the data.

Firstly, the Neural Network generalizes poorly to longer sentences. So, this paper counts the sentence lengths of the datasets, as shown in Table 6.

Additionally, box plots are generated using the data, as shown in Fig. 6. For the SciTail dataset, this paper controls the length of **7** to **24** words, since many rows of this dataset are words or phrases, but this paper researches the similarity of sentences, so the rows with less than 7 words are removed. For the TwitterPPDB dataset, this paper sets the maximum length as **24**, and for the QQP dataset, it is set to **20**.

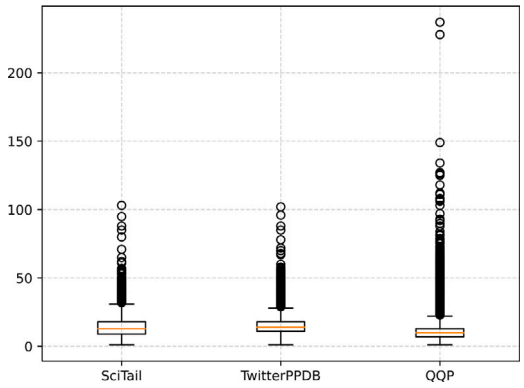

Fig. 6. The box plots of the datasets.

Table 7
The amount of datasets.

Dataset	Positive	Negative	Balance
SciTail	7694	12 377	8078, 8078
TwitterPPDB	11 167	31 033	11 725, 11 725
QQP	149 263	255 027	156 726, 156 726

Table 8
The main training parameters of the three datasets.

Parameter	SciTail	TwitterPPDB	QQP
num_epochs	16	6	2
max_length	24	24	20
batch_size	128	128	128
LSTM_Model_layer1	128	128	128
LSTM_Model_layer2	128	128	128
LSTM_Model_layer3	64	64	64
LSTM_Model_output	128	128	128
CNN_Model_num_filters	128	128	128
CNN_Model_filter_sizes	3,4,5	3,4,5	3,4,5
CNN_Model_output	128	128	128
Optimizer	AdamOptimizer	AdamOptimizer	AdamOptimizer
learning_rate	0.0001	0.0001	0.001

Secondly, this paper uses the Siamese Neural Networks; the number of positive and negative samples needs to be equal. Therefore, this paper uses two methods to maintain the balance of the datasets. One is to ensure that the structure of the original sentence is not destroyed, randomly select 5% of the positive samples, and uses the UniLM [56] algorithm to generate similar sentences to increase the positive samples. Another is randomly negatively sampling the negative samples. These approaches ensure a balance between positive and negative samples, as shown in Table 7.

5.3. Training algorithm and parameters

The training procedure for the TIC-SCNN-SRNN model is detailed in Algorithm 1. The implementation is carried out using TensorFlow, leveraging the placeholder mechanism to dynamically input data while the session is running.

The main training parameters of the three datasets are shown in Table 8. To effectively scale the outputs of SCNN and SRNN, this paper sets their dimensions to 128.

5.4. Accuracy, F1 score, and analyses

This paper uses accuracy and F1 score as evaluation metrics to evaluate the models' performance, calculated using Eqs. (20) and (21).

Algorithm 1 Training the TIC-SCNN-SRNN Model

Require: $\alpha, \beta > 0$: weight for local/context text features
$\eta > 0$: learning rate
Θ^0: initial parameters
Training set $\mathcal{D} = \{(T_i, T_i', y_i)\}_{i=1}^{N}$, $y_i \in \{0, 1\}$
Mini-batch size m
Ensure: Final parameters Θ^*

$t \leftarrow 0$
Initialize Θ^0
while not converged **do**
  $t \leftarrow t + 1$
  Randomly sample a mini-batch $\mathcal{B} = \{i_1, \ldots, i_m\}$ from $\mathcal{D}$
  for $k = 1 \rightarrow m$ **do**
    $\text{SCNN\_out}^{(k)} \leftarrow f_{\text{SCNN}}(T_{i_k}, T'_{i_k}; \Theta_{\text{SCNN}})$
    $\text{SRNN\_out}^{(k)} \leftarrow f_{\text{SRNN}}(T_{i_k}, T'_{i_k}; \Theta_{\text{SRNN}})$
    $\hat{y}_{\text{SCNN}}^{(k)} \leftarrow \sigma\big(W_{\text{SCNN}} \cdot \text{SCNN\_out}^{(k)} + b_{\text{SCNN}}\big)$
    $\hat{y}_{\text{SRNN}}^{(k)} \leftarrow \sigma\big(W_{\text{SRNN}} \cdot \text{SRNN\_out}^{(k)} + b_{\text{SRNN}}\big)$
    $\text{Loss}_1^{(k)} \leftarrow -\big[y_{i_k} \log(\hat{y}_{\text{SCNN}}^{(k)}) + (1 - y_{i_k}) \log(1 - \hat{y}_{\text{SCNN}}^{(k)})\big]$
    $\text{Loss}_2^{(k)} \leftarrow -\big[y_{i_k} \log(\hat{y}_{\text{SRNN}}^{(k)}) + (1 - y_{i_k}) \log(1 - \hat{y}_{\text{SRNN}}^{(k)})\big]$
    $\text{Loss}^{(k)} \leftarrow \alpha\, \text{Loss}_1^{(k)} + \beta\, \text{Loss}_2^{(k)}$
  end for
  $\mathcal{L}_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{k=1}^{m} \text{Loss}^{(k)}$
  $\Theta^t \leftarrow \Theta^{t-1} - \eta\, \nabla_{\Theta}\, \mathcal{L}_{\mathcal{B}}(\Theta^{t-1})$
end while
return Θ^*

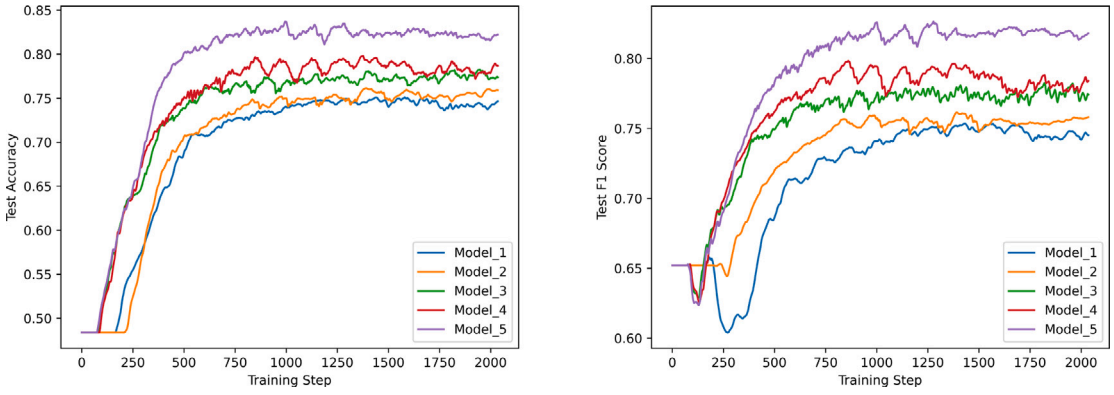

Fig. 7. The accuracy and F1 scores of SciTail dataset.

$$\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{20}$$

$$F1 = \frac{2 \times \frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} \tag{21}$$

In Eqs. (20) and (21), *TP* is the true positive, which means successfully predicting positive samples as positive samples. *TN* is the true negative, which means successfully predicting negative samples as negative samples; *FP* is the false positive, which means mispredicting negative samples as positive samples; and *FN* is the false negative, which means mispredicting positive samples as negative samples.

This paper experiments on Model_1, Model_2, Model_3, Model_4, and model_5 using SciTail, TwitterPPDB, and QQP, respectively. These three datasets' accuracy and F1 scores are shown in Figs. 7, 8, and 9, with the corresponding values presented in Table 9.

It can be seen from Figs. 7, 8, and 9. Model_5 exhibits the highest accuracy and F1 score. Model_3 and Model_4 demonstrate higher accuracy and F1 scores compared to Model_1 and Model_2. Additionally, Model_2 outperforms Model_1 in both accuracy and F1 score, while Model_4 exceeds Model_3 in the same metrics.

Combine the accuracy and F1 scores of these models; the following inequalities can be obtained:

$$\begin{cases} Acc._{Model\_1} < Acc._{Model\_2} \\ F1_{Model\_1} < F1_{Model\_2} \end{cases} \tag{22}$$

$$\begin{cases} Acc._{Model\_3} < Acc._{Model\_4} \\ F1_{Model\_3} < F1_{Model\_4} \end{cases} \tag{23}$$

$$\begin{cases} (Acc._{Model\_5} - Acc._{Model\_4}) < (Acc._{Model\_5} - Acc._{Model\_2}) \\ (F1_{Model\_5} - F1_{Model\_4}) < (F1_{Model\_5} - F1_{Model\_2}) \end{cases} \tag{24}$$

According to **Inequalities** (22)–(24), the following conclusions can be drawn:

1. From **Inequalities** (22), it can be concluded that SRNN is effective in text similarity tasks. Model_2 set $\beta = 0.000001$ and Model_1 set $\beta = 0$ indicate that the SRNN in Model_1 is not working, while the SRNN in Model_2 is active. That is to say, Model_2 has improved the accuracy and F1 score by adding SRNN. It shows that SRNN is effective for text similarity tasks. Moreover, this paper uses three Bidirectional LSTM layers for the SRNN, long-term dependency can be learned better, it can extract context text features more effectively.

2. From **Inequalities** (23), it is evident that SCNN is effective in text similarity tasks. Model_4 set $\alpha = 0.000001$ and Model_3 set

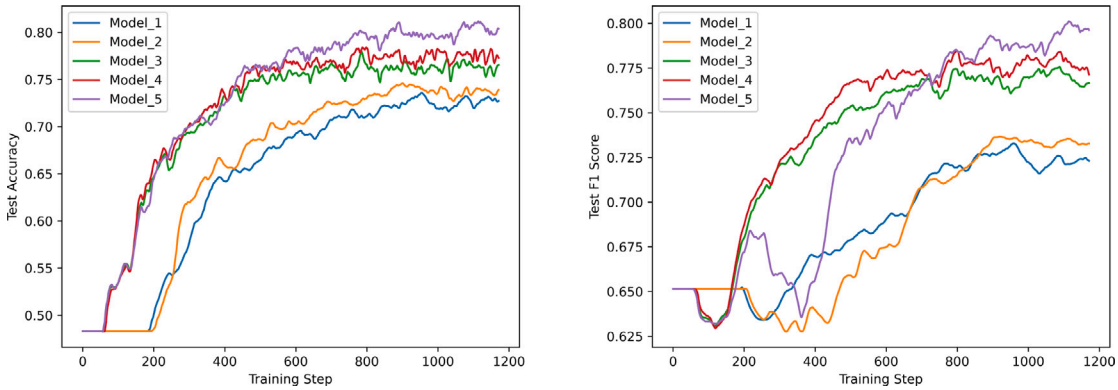

Fig. 8. The accuracy and F1 scores of TwitterPPDB dataset.

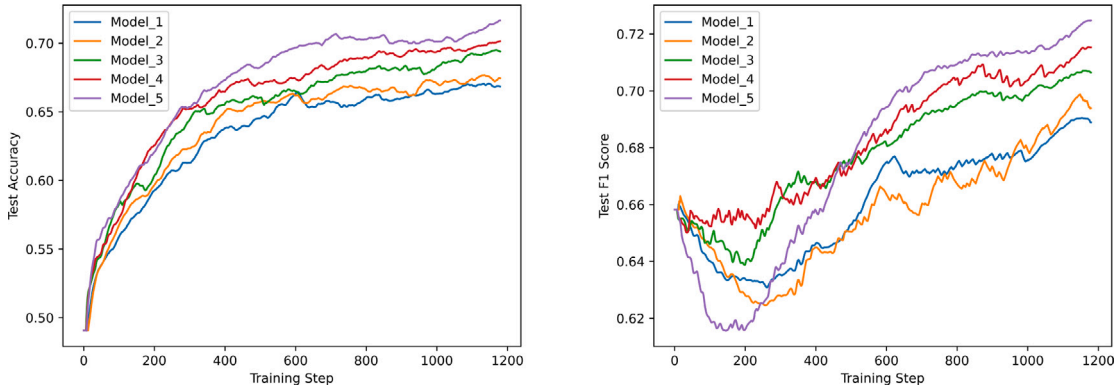

Fig. 9. The accuracy and F1 scores of QQP dataset.

Table 9
The values of the accuracy and F1 scores for SciTail, TwitterPPDB, and QQP datasets.

Name	SciTail Acc.	SciTail F1	TwitterPPDB Acc.	TwitterPPDB F1	QQP Acc.	QQP F1
Model_1	0.748	0.745	0.728	0.723	0.668	0.688
Model_2	0.760	0.759	0.736	0.733	0.675	0.690
Model_3	0.774	0.774	0.760	0.768	0.694	0.708
Model_4	0.788	0.784	0.771	0.771	0.703	0.722
Model_5	0.823	0.820	0.801	0.797	0.719	0.731

$\alpha = 0$ indicate that the SCNN in Model_3 is not working, while the SCNN in Model_4 is active. This demonstrates that the inclusion of SCNN in Model_4 improves accuracy and the F1 score, confirming the effectiveness of SCNN. Furthermore, this paper uses TextCNN for the SCNN, the TextCNN contains multiple filter sizes. This model can better extract the feature of a phrase (3 to 5 words); that is, it can extract local text features more effectively.

3. From **Inequalities** (22) and (23), it can be inferred that combining SCNN and SRNN improves text similarity accuracy and F1 score. Model_2 outperforms Model_1 due to the addition of SRNN, and Model_4 outperforms Model_3 due to the addition of SCNN. Because both Model_2 and Model_4 include SCNN and SRNN, these models can effectively extract local and context text features.

4. From **Inequalities** (24), it becomes clear that SRNN is more effective than SCNN in text similarity tasks. Compared with Model_4, Model_5 has increased the weight of SCNN, and compared with Model_2, Model_5 has increased the weight of SRNN. The improved accuracy and F1 score of Model_5 relative to Model_2 are much higher than that of Model_4. In other words, increasing the weight of SRNN can effectively improve the model's performance. Because for text data, context text features are more important than local text features. The context features contain the meaning of the entire sentence. In contrast, the local text features only capture the features of phrases, making it difficult to describe the meaning of the entire sentence.

In summary, the experimental results show that both the SRNN and the SCNN are effective for text similarity tasks, but the SRNN is more effective than the SCNN.

5.5. Comparison with other CNN&RNN algorithms

In order to verify the advantages of the TIC-SCNN-SRNN model, this paper tests the accuracy and F1 score of the state-of-the-art CNN&RNN models on the SciTail, TwitterPPDB, and QQP datasets, respectively. The test results are shown in Table 10.

Model_A uses the output of CNN as part of RNN's input; the main contribution of this model comes from LSTM. This model extracts text features weaker than others, so the accuracy and F1 score are the lowest.

Model_B, **Model_C**, **Model_D**, and **Model_E** are all sequence models. The accuracy and F1 scores of Model_D and Model_E are higher than that of Model_B and Model_C. These improvements are attributed to the RNN in Model_D and Model_E directly processing the original texts, allowing for better extraction of context text features. In contrast, the CNN in Model_B and Model_C, which directly process the original texts, lead to the loss of context text features.

Model_B exhibits lower accuracy and F1 score than Model_C because Model_B uses TF-IDF to extract features, leading to the loss of

Table 10
The accuracy and F1 score of other CNN&RNN algorithms.

Name	Model	SciTail	TwitterPPDB	QQP
Model_A	Siamese CNN&LSTM [18]	0.792, 0.793	0.777, 0.774	0.706, 0.722
Model_B	TF-IDF and CNN-LSTM [21]	0.795, 0.794	0.779, 0.775	0.707, 0.723
Model_C	CNN-LSTM [22]	0.797, 0.796	0.780, 0.777	0.708, 0.723
Model_D	LSTM-CNN [23]	0.803, 0.799	0.784, 0.780	0.708, 0.725
Model_E	LDA and BiLSTM-CNN [20]	0.805, 0.800	0.785, 0.781	0.709, 0.725
Model_F	Combined-RNN-CNN [19]	0.811, 0.809	0.790, 0.789	0.712, 0.727
Model_5	TIC-SCNN-SRNN	0.823, 0.820	0.801, 0.797	0.719, 0.731

some original text features. However, Model_B can be trained faster and has fewer parameters.

Model_E achieves higher accuracy and F1 score compared to Model_D. This improvement is attributed to Model_E, which uses LDA to expand the features of short texts.

Model_F performs higher accuracy and F1 score than Model_E. Because Model_F has two channels, CNN and RNN, the extraction of text features is better. Nevertheless, it does not perform as well as Model_5 because the two channels of this model are not independent; it cannot reflect the difference between CNN and RNN in text features extractions.

Model_5 reaches the highest accuracy and F1 score on the test set, indicating that the model proposed in this paper is more effective for the text similarity tasks. This superiority is attributed to the model's two independent channels, SCNN and SRNN. SCNN excels at extracting local text features, while SRNN is more effective at capturing context text features. These two channels do not affect each other, so the TIC-SCNN-SRNN model can fully extract the local and context features of the texts.

6. Conclusion

This paper first expounds on the basic definition of text similarity and explains local and context text features. Secondly, a literature review is conducted on text similarity, including traditional and deep learning algorithms. Moreover, briefly analyze the shortcomings of these algorithms. Thirdly, the methodology for text similarity based on Siamese Neural Networks is expounded. Fourth, the TIC-SCNN-SRNN model is elaborated, mainly including the architecture of TIC-SCNN-SRNN, SCNN, and SRNN. Finally, the proposed models have experimented on SciTail, TwitterPPD, and QQP datasets. The experimental results show that the SCNN and the SRNN are effective for the text similarity tasks, but the SRNN is more effective than the SCNN. Furthermore, it is demonstrated that the model proposed in this paper is state-of-the-art when compared with other CNN&RNN models.

The key findings of this paper are as follows:

1. The Siamese Convolutional Neural Networks (SCNN) model is proposed to address the problem of insufficient extraction of local text features.
2. The Siamese Recurrent Neural Networks (SRNN) model is proposed to address the problem of insufficient extraction of context text features.
3. The Two Independent Channels: Siamese Convolutional Neural Networks and Siamese Recurrent Neural Networks (TIC-SCNN-SRNN) model is proposed to address the problem of no distinction between the effectiveness of local and context text features extractions.

By leveraging independent channels for local and context text features extractions, the TIC-SCNN-SRNN model demonstrates superior performance in text similarity tasks. This novel framework effectively addresses critical challenges in this domain. These findings lay the groundwork for innovative developments in related fields. For instance, in database management, it can detect redundant entries, thereby optimizing storage. In information retrieval, the model can filter search results to identify the most relevant responses to user queries, thereby improving retrieval accuracy. Additionally, in question-answering systems, it can evaluate the similarity between user questions and stored answers, enhancing response precision. Furthermore, the model's ability to cluster related documents makes it invaluable for text clustering, enabling the efficient categorization of large datasets.

Several limitations should be acknowledged:

1. This study employs Word2Vec as the pre-training model to generate word embeddings. However, recent advancements in NLP have introduced state-of-the-art models, such as BERT [38] and OpenAI's GPT [57], which leverage context embeddings to achieve superior performance.
2. The current study adopts a binary classification framework, where text pairs are labeled as either similar (1) or dissimilar (0). While effective, this framework is limited in its capacity to represent nuanced degrees of similarity.
3. The experiments conducted in this study focus on three datasets: SciTail, TwitterPPD, and QQP. Although these datasets are suitable for benchmarking, they may not fully encompass the diversity of real-world scenarios in which text similarity models are utilized.

Future research directions include:

1. Integrating an advanced pre-trained model (such as BERT [38] or OpenAI's GPT [57]) to further enhance the performance of text similarity tasks.
2. Extending the model's output to a continuous scale (e.g., from 0 to 5) to capture finer-grained levels of similarity, thereby enabling more precise and practical applications.
3. Expanding experiments to larger and more diverse datasets across multiple domains to further validate and generalize the model's effectiveness.

In conclusion, by independently addressing the extraction of local and context text features, the TIC-SCNN-SRNN model offers a novel and effective solution for text similarity tasks and sets a promising foundation for future research in this area.

Declaration of competing interest

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

Data availability

Data will be made available on request.

References

[1] L.A. Henry, Information search strategies on the internet: A critical component of new literacies, Webology 2 (1) (2005) 1–11.

[2] T. Willoughby, S.A. Anderson, E. Wood, J. Mueller, C. Ross, Fast searching for information on the internet to use in a learning context: The impact of domain knowledge, Comput. Educ. 52 (3) (2009) 640–648.

[3] W.H. Gomaa, A.A. Fahmy, A survey of text similarity approaches, Int. J. Comput. Appl. 68 (13) (2013) 13–18.

[4] S. Butakov, S. Murzintsev, A. Tskhai, Detecting text similarity on a scalable no-SQL database platform, in: 2016 International Conference on Platform Technology and Service (PlatCon), 2016, pp. 1–5.

[5] D. Bollegala, Y. Matsuo, M. Ishizuka, Measuring semantic similarity between words using web search engines, Www 7 (2007) (2007) 757–766.

[6] F. Bravo-Marquez, G. L'Huillier, S.A. Ríos, J.D. Velásquez, A text similarity meta-search engine based on document fingerprints and search results records, in: 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Vol. 1, 2011, pp. 146–153.

[7] S. Minaee, Z. Liu, Automatic question-answering using a deep similarity neural network, in: 2017 IEEE Global Conference on Signal and Information Processing, GlobalSIP, 2017, pp. 923–927.

[8] S.G. Aithal, A.B. Rao, S. Singh, Automatic question-answer pairs generation and question similarity mechanism in question answering system, Appl. Intell. 51 (11) (2021) 8484–8497.

[9] A. Huang, Similarity measures for text document clustering, in: Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand, Vol. 4, 2008, pp. 9–56.

[10] D. Lin, An information-theoretic definition of similarity, in: Icml, Vol. 98, 1998, pp. 296–304.

[11] W. Hu, A. Dang, Y. Tan, A survey of state-of-the-art short text matching algorithms, in: Data Mining and Big Data: 4th International Conference, DMBD 2019, Chiang Mai, Thailand, July 26–30, 2019, Proceedings 4, Springer, 2019, pp. 211–219.

[12] A. Severyn, A. Moschitti, Learning to rank short text pairs with convolutional deep neural networks, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Association for Computing Machinery, 2015, pp. 373–382.

[13] H. He, K. Gimpel, J. Lin, Multi-perspective sentence similarity modeling with convolutional neural networks, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1576–1586.

[14] J. Mueller, A. Thyagarajan, Siamese recurrent architectures for learning sentence similarity, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI Press, 2016, pp. 2786–2792.

[15] P. Neculoiu, M. Versteegh, M. Rotaru, Learning text similarity with siamese recurrent networks, in: Proceedings of the 1st Workshop on Representation Learning for NLP, 2016, pp. 148–157.

[16] J.V.A. de Souza, L.E.S.E. Oliveira, Y.B. Gumiel, D.R. Carvalho, C.M.C. Moro, Exploiting siamese neural networks on short text similarity tasks for multiple domains and languages, in: International Conference on Computational Processing of the Portuguese Language, Springer, 2020, pp. 357–367.

[17] C. Lv, F. Wang, J. Wang, L. Yao, X. Du, Siamese multiplicative LSTM for semantic text similarity, in: Proceedings of the 2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence, 2020, pp. 1–5.

[18] E. Linhares Pontes, S. Huet, A.C. Linhares, J.-M. Torres-Moreno, Predicting the semantic textual similarity with siamese CNN and LSTM, in: Traitement Automatique des Langues Naturelles, TALN, 2018, pp. 311–319.

[19] M.S.H. Ameur, R. Belkebir, A. Guessoum, Robust arabic text categorization by combining convolutional and recurrent neural networks, ACM Trans. Asian Low-Resour. Lang. Inf. Process. (TALLIP) 19 (5) (2020) 1–16.

[20] X. Niu, W. Zheng, Y. Xiao, Q. Wang, Short text similarity computation method based on feature expansion and Siamese network, in: 2021 4th International Conference on Data Science and Information Technology, 2021, pp. 268–272.

[21] H. Zhou, Research of text classification based on TF-IDF and CNN-LSTM, in: Journal of Physics: Conference Series, Vol. 2171, IOP Publishing, 2022, 012021.

[22] L.B.S. Balita, K.M.A. Degrano, A.D.A. Pamoso, J.C. De Goma, Sentiment analysis on book reviews using convolutional neural network (CNN) long short-term memory (LSTM) hybrid, in: Proceedings of the 2022 6th International Conference on E-Business and Internet, 2022, pp. 125–132.

[23] J. Zhang, Y. Li, J. Tian, T. Li, LSTM-CNN hybrid model for text classification, in: 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference, IAEAC, IEEE, 2018, pp. 1675–1680.

[24] Z. He, C.E. Dumdumaya, V. Quimno, Measurement of semantic text similarity, J. Theor. Appl. Inf. Technol. 102 (5) (2024) 1673–1685.

[25] E.S. Ristad, P.N. Yianilos, Learning string-edit distance, IEEE Trans. Pattern Anal. Mach. Intell. 20 (5) (1998) 522–532.

[26] X. Xu, L. Chen, P. He, Fast sequence similarity computing with LCS on LARPBS, in: International Symposium on Parallel and Distributed Processing and Applications, Springer, 2005, pp. 168–175.

[27] G. Kondrak, N-gram similarity and distance, in: International Symposium on String Processing and Information Retrieval, Springer, 2005, pp. 115–126.

[28] S. Niwattanakul, J. Singthongchai, E. Naenudorn, S. Wanapu, Using of Jaccard coefficient for keywords similarity, in: Proceedings of the International Multiconference of Engineers and Computer Scientists, Vol. 1, 2013, pp. 380–384.

[29] Z. Harris, Distributional hypothesis, Word World 10 (23) (1954) 146–162.

[30] G. Salton, A. Wong, C.-S. Yang, A vector space model for automatic indexing, Commun. ACM 18 (11) (1975) 613–620.

[31] T.K. Landauer, S.T. Dumais, A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge, Psychol Rev 104 (2) (1997) 211–240.

[32] T. Hofmann, Probabilistic latent semantic indexing, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999, pp. 50–57.

[33] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (Jan) (2003) 993–1022.

[34] G.A. Miller, Wordnet: a lexical database for english, Commun. ACM 38 (11) (1995) 39–41.

[35] M. Strube, S.P. Ponzetto, WikiRelate! computing semantic relatedness using wikipedia, in: AAAI, Vol. 6, 2006, pp. 1419–1424.

[36] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using Wikipedia-based explicit semantic analysis, in: IJcAI, Vol. 7, 2007, pp. 1606–1611.

[37] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, L. Heck, Learning deep structured semantic models for web search using clickthrough data, in: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, 2013, pp. 2333–2338.

[38] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.

[39] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, ALBERT: A lite BERT for self-supervised learning of language representations, in: International Conference on Learning Representations, 2020, pp. 1–17.

[40] P. He, X. Liu, J. Gao, W. Chen, DeBERTa: Decoding-enhanced BERT with disentangled attention, in: International Conference on Learning Representations, 2021, pp. 1–23.

[41] H. Zhengfang, I.K.D. Machica, B. Zhimin, Textual similarity based on double siamese text convolutional neural networks and using BERT for pre-training model, in: 2022 5th International Conference on Artificial Intelligence and Big Data, ICAIBD, IEEE, 2022, pp. 107–111.

[42] Z. He, C.E. Dumdumaya, Text similarity based on siamese multi-output bidirectional long short-term memory, in: 2024 6th International Conference on Natural Language Processing, ICNLP, IEEE, 2024, pp. 51–55.

[43] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a "Siamese" time delay neural network, in: Proceedings of the 6th International Conference on Neural Information Processing Systems, Morgan Kaufmann Publishers Inc., 1993, pp. 737–744.

[44] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'05, Vol. 1, IEEE, 2005, pp. 539–546.

[45] Z. He, W. Su, Z. Bi, M. Wei, Y. Dong, G. Xu, The improved siamese network in face recognition, in: 2019 International Conference on Intelligent Computing, Automation and Systems, ICICAS, IEEE, 2019, pp. 443–446.

[46] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, Curran Associates Inc., 2013, pp. 3111–3119.

[47] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, Association for Computational Linguistics, 2014, pp. 1746–1751.

[48] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.

[49] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[50] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.

[51] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nat. 529 (7587) (2016) 484–489.

[52] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., {TensorFlow}: a system for {Large-Scale} machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 16, 2016, pp. 265–283.

[53] T. Khot, A. Sabharwal, P. Clark, Scitail: A textual entailment dataset from science question answering, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018, pp. 5189–5197.

[54] W. Lan, S. Qiu, H. He, W. Xu, A continuously growing dataset of sentential paraphrases, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 1224–1234.
[55] Z. Wang, W. Hamza, R. Florian, Bilateral multi-perspective matching for natural language sentences, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, AAAI Press, 2017, pp. 4144–4150.
[56] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, H.-W. Hon, Unified language model pre-training for natural language understanding and generation, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., 2019, pp. 13063–13075.
[57] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training, 2018, pp. 1–12.

Zhengfang He, Doctor of Information Technology, has research interests in artificial intelligence, deep learning, and natural language processing.