ERCNN: Enhanced Recurrent Convolutional Neural Networks for Learning Sentence Similarity

Niantao Xie[1], Sujian Li[1,2], and Jinglin Zhao[3]

[1] MOE Key Laboratory of Computational Linguistics, Peking University, China
{xieniantao,lisujian}@pku.edu.cn
[2] Peng Cheng Laboratory, Shenzhen, China
[3] Faculty of Arts and Social Science, National University of Singapore, Singapore
zhaojinglin92@gmail.com

Abstract. Learning the similarity between sentences is made difficult by the fact that two sentences which are semantically related may not contain any words in common limited to the length. Recently, there have been a variety kind of deep learning models which are used to solve the sentence similarity problem. In this paper we propose a new model which utilizes enhanced recurrent convolutional neural network (ERCNN) to capture more fine-grained features and the interactive effects of key-points in two sentences to learn sentence similarity. With less computational complexity, our model yields state-of-the-art improvement compared with other baseline models in paraphrase identification task on the Ant Financial competition dataset.

Keywords: Sentence Similarity · ERCNN · Soft attention mechanism.

1 Introduction

Paraphrase identification is an important NLP task which may be greatly enhanced by modeling the underlying semantic similarity between compared texts. In particular, a good method should not be susceptible to variations of wording or syntax used to express the same idea, and also should be able to measure similarity for both long and short texts like sentences.

Learning such sentences similarity has attracted many research interests[17]. However, it still remains a knotty issue. With the renaissance of deep learning, many text representation methods are combined to take better into account both semantics and structure of sentences for paraphrase identification task. Text representation is the key component in measuring semantic similarity and usually composed of convolutional neural network (CNN)[14] and recurrent neural network (RNN). CNN is supposed to be better at extracting robust and abstract features of texts while RNN, especially the Long Short-Term Memory (LSTM) [11] and Gated Recurrent Unit (GRU) [6] have been applied widely in paraphrase identification task because of their naturally suited for variable-length inputs.

Some prior work also proposes attention-based CNN or RNN models to focus on the mutual influence of two sentences in text representation [5, 21, 23].

Following common solutions in paraphrase identification task, prior works have demonstrated the effectiveness of two types of deep learning frameworks. The first framework is based on the "Siamese network" [3, 17, 23]. And the second framework is called "matching-aggregation" [5, 22]. The idea behind these two frameworks is to measure semantic similarity from multiple perspectives by capturing interactive features between the text representations of input sentences. And the difference between them is mainly on the way of capturing interactive features. "Siamese network" framework simply concatenates on the text representations of input sentences by calculating their cosine or other distances. Due to its simple way of capturing interactive features, "Siamese network" framework becomes easy to train. Conversely, "matching-aggregation" framework makes more improvements on interactive layer by utilizing a complex way to capture the fine-grained features and therefore performs better than "Siamese network" framework usually.

In this paper, we introduce our model based on one state-of-the-art work named ESIM[5]to measure sentence similarity by jointly leveraging the model predictive effect and parameter size. As a result, the improvements on text representation and optimization of capturing interactive features are main contributions of our model. The details are as follows:

- Different from ESIM, we add CNN layers upon RNN layers to reduce the parameters and capture more fine-grained features during input encoding.
- Moreover, we further simplify ESIM structure by using fully-connected layers to replace the time-consuming BiLSTM layers in overall similarity modeling. Based on the methods above, we achieve additional improvements in paraphrase identification task on Ant Financial competition dataset.

2 Related Work

The deep learning applications for paraphrase identification task have recently received much attention [17], from the availability of high-quality semantic word representations [7, 16, 20] and followed by the seminal papers introduce "Siamese network" and "matching-aggregation" framework for learning sentence similarity. In the past few years, many deep learning models based on "Siamese network" or "matching-aggregation" framework have made progress in learning sentence similarity [5, 17, 22, 23].

For "Siamese network" framework [3, 17, 19, 23], two input sentences are applied by the same neural network encoder like a CNN or a RNN individually, so both of the two sentences are encoded into vectors of the same embedding space. As a result, a matching interaction is calculated solely based on the two sentence vectors. The advantage of this framework is that sharing parameters makes the model simply equipped and therefore effortless to train, and the sentence vectors can be used directly for measuring similarity based on cosine or other type of distances. However, there is no explicit interaction between the two sentences during the encoding procedure, which may lose some crucial information.

To deal with this problem, some prior works also incorporate attention mechanism into "Siamese network" framework[21, 23]to address the problem of insufficient interactions. Different from learning each sentence's representation separately, the attention-based models consider the mutual influence between two sentences. One of famous models is ABCNN [23], which is an attention-based CNN for modeling sentence similarity.

For "matching-aggregation" framework [5, 22], smaller units such as words or characters of the two sentences are firstly matched, while the prior works usually employ bidirectional RNN to encode variable-length sentences into a fixed-length vector, and then the matching results are usually aggregated by RNN into a vector to make the final decision. This framework captures more interactive features between the two sentences, therefore significant improvements are obtained naturally. However, this framework still remains some defects, and the main shortcoming exists in the time-consuming matching operation in capturing interactive features.

One famous model based on "matching-aggregation" framework is called ESIM[5], which achieves the state-of-the-art performance in many NLP tasks besides sentence similarity by employing attention-based LSTM to capture some high-order interactions between two sentences. Motivated by the idea of ESIM, a recently proposed model named BiMPM (bilateral multi-perspective matching) [22] emerges. BiMPM brings in many complex matching operations to enhance the model ability to extract mutual information between two sentences.

As introduced before, our work is also related to ESIM using attention-based RNN for paraphrase identification task[5]. But unlike other models, we incorporate CNN to process the RNN output and simplify the time-consuming recursive architectures by fully-connected layers.

3 ERCNN Model

Assume we have two sentences $\boldsymbol{a} = (\boldsymbol{a}_1, ..., \boldsymbol{a}_{l_a})$ and $\boldsymbol{b} = (\boldsymbol{b}_1, ..., \boldsymbol{b}_{l_b})$, where $\boldsymbol{a}_i$ or $\boldsymbol{b}_j \in R^k$ is an embedding of k-dimensional vector, which can be initialized with some pre-trained word or character embeddings. The goal is to predict a label $\mathbf{y}$ that indicates the similarity between $\boldsymbol{a}$ and $\boldsymbol{b}$.

Our ERCNN model is composed of three components: (①) input encoding, (②) local similarity modeling, and (③) overall similarity modeling. These components are explained in details in the following sections. Here the input sentences are preprocessed respectively in word-level and character-level, and they are trained separately and final predictions are the average of two outputs. The left part of figure 1 shows a high-level view of the architecture.

3.1 Input Encoding

BiGRU We first employ bidirectional GRU (BiGRU)[6] to encode the two input sentences (Equation (1) and (2)). Here BiGRU learns to represent a word

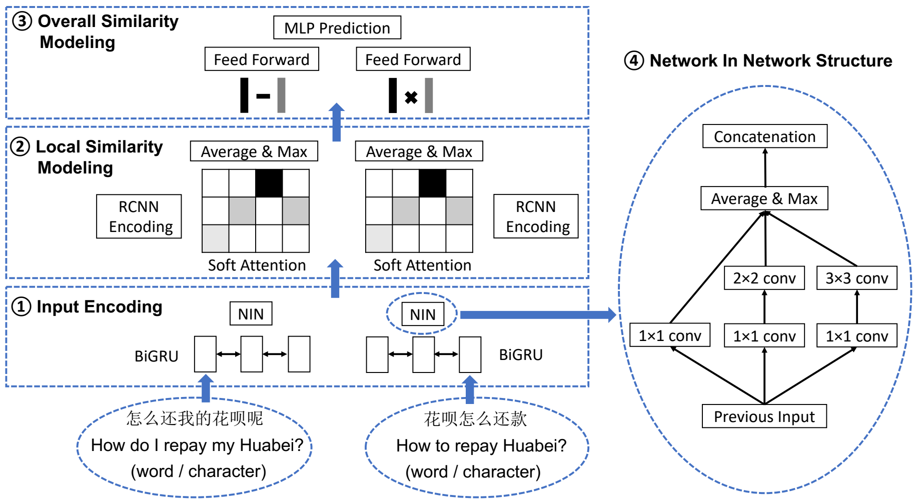

Fig. 1. The architecture of our introduced model.

or character and its corresponding hidden layer state as follows:

$$\overline{\boldsymbol{a}}_i = BiGRU(\boldsymbol{a}, i), \forall i \in [1, ..., l_a]. \tag{1}$$

$$\overline{\boldsymbol{b}}_j = BiGRU(\boldsymbol{b}, j), \forall j \in [1, ..., l_b]. \tag{2}$$

BiGRU contains two GRUs starting from the left and the right side respectively. The hidden states generated by these two GRUs at each time step are concatenated to represent the status of time step.

Note that we make use of GRU memory blocks in our models instead of LSTMs in original ESIM model. We compared LSTM[11] with GRU in this part and LSTMs are inferior to GRUs on results of experiments for this task.

NIN The most difference between our model and ESIM is that we apply CNN to aggregate the input encoding after the process of BiGRU. Because CNN and RNN have different mechanism of action in sentence representation: CNN performs better in extracting key words from sentence while RNN behaves preferable in extracting sequence information in the sentence. Therefore, by incorporating both the merits of CNN and RNN, we can capture more fine-grained features during input encoding. Moreover, due to the peculiarity of parameter sharing, CNN can help reduce the number of parameters in the model dramatically. In the next part of experiment, we will prove our improvements by detailed examples.

Here we refer the idea of "Network In Network" (NIN)[15, 10] which is illustrated as (④) network in network structure in the right part of figure 1 to design our CNN layers. The first layer contains three convolutions of size 1×1

and the second layer consists of two convolutions of sizes 2×2 and 3×3. The 1×1 convolution operations help reduce the dimension of output feature map and increase non-linearity by the active function Relu[18], and the 2×2 and 3×3 convolution operations here further extract robust and abstract features based on the output of BiGRU layer.

Let's take the convolution operation on BiGRU layer for example, the BiGRU output $\overline{\boldsymbol{p}} \in R^{n \cdot d_0}$ which is a sentence of length n (padded where necessary) is represented as:

$$\overline{\boldsymbol{p}}_{1:n} = \overline{\boldsymbol{p}}_1 \oplus \overline{\boldsymbol{p}}_2 \oplus \overline{\boldsymbol{p}}_3 \oplus ... \oplus \overline{\boldsymbol{p}}_n \tag{3}$$

where $\oplus$ is the concatenation operator and d_0 is the hidden state size. Let $\overline{\boldsymbol{p}}_{i:i+j}$ refer to the concatenation of hidden states $\overline{\boldsymbol{p}}_i, \overline{\boldsymbol{p}}_{i+1}, ..., \overline{\boldsymbol{p}}_{i+j}$. The convolutional operation involves a filter $\mathbf{W}_f \in R^{w \cdot d_0}$ which is applied to a window of w words to product a new feature. For instance, a feature $\boldsymbol{p}_i$ is generated from a window of words $\overline{\boldsymbol{p}}_{i:i+w-1}$ as the following formula:

$$p_i = ReLU(\mathbf{W}_f \cdot \ \overline{\boldsymbol{p}}_{i:i+w-1} + b) \tag{4}$$

here $b \in R$ is a bias term. Because the window may be outside of the sentence boundaries when it slides near the boundary, we set special padding tokens for the sentence, which means that we take the out-of-range input vectors $\overline{\boldsymbol{p}}_i$ (i < 1 or i $> n$) as zero vectors.

This filter is applied to each possible window of hidden states in the sentence $\{\overline{\boldsymbol{p}}_{1:w}, \overline{\boldsymbol{p}}_{2:w+1}, ..., \overline{\boldsymbol{p}}_{n-w+1:n}\}$ to produce a feature map:

$$\boldsymbol{p} = [p_1, p_2, ..., p_{n-w+1}] \tag{5}$$

where $\boldsymbol{p} \in R^{d_1}$ and $d_1 = n - w + 1$.

As a whole, we acquire the NIN encoding in the following way:

$$\widetilde{\boldsymbol{p}} = NIN(\overline{\boldsymbol{p}}) \tag{6}$$

Pooling Pooling is commonly used to extract features from convolution output. In this part, we employ column-wise average and max pooling, and this architecture supports to stack an arbitrary number of convolution-pooling blocks to extract increasingly abstract features.

3.2 Local Similarity Modeling

Modeling local subsentential similarity between two inputs is the basic component to determine the overall similarity. It needs to employ some forms of alignment[1] to associate the relevant parts between two input sentences. Here we use soft attention to achieve the alignment. Our soft alignment layer computes the attention weights as the similarity of two sentences encoding with Equation (4).

$$\boldsymbol{e}_{ij} = \overline{\boldsymbol{p}}_{ai}^T \cdot \overline{\boldsymbol{p}}_{bj} \tag{7}$$

where $\overline{\boldsymbol{p}}_{ai}$ and $\overline{\boldsymbol{p}}_{bj}$ are the i^{th} BiGRU output of sentence $\boldsymbol{a}$ and the j^{th} BiGRU output of sentence $\boldsymbol{b}$ separately. Local similarity is measured by the attention weight e_{ij} computed above. For the encoding of a word or character in one sentence, i.e., $\overline{\boldsymbol{p}}_{ai}$, the relevant semantics in another sentence is measured by e_{ij} specifically with Equation (8).

$$\hat{\boldsymbol{p}}_{ai} = \sum_{j=1}^{l_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_b}\exp(e_{ik})}\overline{\boldsymbol{p}}_{bj}, \forall i \in [1, ..., l_a] \tag{8}$$

$$\hat{\boldsymbol{p}}_{bj} = \sum_{i=1}^{l_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_a}\exp(e_{kj})}\overline{\boldsymbol{p}}_{ai}, \forall j \in [1, ..., l_b] \tag{9}$$

where $\hat{\boldsymbol{p}}_{ai}$ is a weighted summation of $\{\overline{\boldsymbol{p}}_{bj}\}_{j=1}^{l_b}$. Intuitively, the content in $\{\overline{\boldsymbol{p}}_{bj}\}_{j=1}^{l_b}$ that is relevant to $\overline{p}_{ai}$ will be selected and represented as $\hat{\boldsymbol{p}}_{ai}$. The same process is performed as $\hat{\boldsymbol{p}}_{bj}$ with Equation (9).

The weighted vectors obtained above are processed with pooling layer. We employ the following strategy: compute both average and max pooling, and concatenate all these vectors with NIN outputs to form the final fixed-length vectors $\boldsymbol{o}_a$ and $\boldsymbol{o}_b$. The process of calculation is as follows:

$$\hat{\boldsymbol{p}}_{a,ave} = \sum_{i=1}^{l_a} \frac{\hat{\boldsymbol{p}}_{a,i}}{l_a},\ \hat{\boldsymbol{p}}_{a,max} = \max_{i=1}^{l_a} \hat{\boldsymbol{p}}_{a,i} \tag{10}$$

$$\hat{\boldsymbol{p}}_{b,ave} = \sum_{j=1}^{l_b} \frac{\hat{\boldsymbol{p}}_{b,j}}{l_b},\ \hat{\boldsymbol{p}}_{b,max} = \max_{j=1}^{l_b} \hat{\boldsymbol{p}}_{b,j} \tag{11}$$

$$\boldsymbol{o}_a = [\hat{\boldsymbol{p}}_{a,ave}; \widetilde{\boldsymbol{p}}_a; \hat{\boldsymbol{p}}_{a,max}] \tag{12}$$

$$\boldsymbol{o}_b = [\hat{\boldsymbol{p}}_{b,ave}; \widetilde{\boldsymbol{p}}_b; \hat{\boldsymbol{p}}_{b,max}] \tag{13}$$

3.3 Overall Similarity Modeling

To determine the overall similarity between two sentences, we design a concatenation layer to perform the similarity modeling. This layer has two type of concatenation. The first concatenation includes the difference and the element-wise product for the interactive sentence representation. And second concatenation further concatenates the output of the first concatenation. The process is specified as follows:

$$\boldsymbol{m}_{out} = [\boldsymbol{o}_a - \boldsymbol{o}_b; \boldsymbol{o}_a \odot \boldsymbol{o}_b] \tag{14}$$

This process could be regarded as a special case of modeling some high-order interaction between the tuple elements. We then put $\boldsymbol{m}_{out}$ into a final multilayer perceptron (MLP) classifier:

$$\mathbf{y} = MLP(\boldsymbol{m}_{out}) \tag{15}$$

This is also a main difference between our ERCNN and ESIM. ESIM here still applies two BiLSTM layers to perform overall similarity modeling but BiLSTM is time-consuming in training process. Based on the feedback of experimental results, here we employ MLP to replace original BiLSTM and achieve the similar results with the number of parameters reduced.

4 Experiments

4.1 Data Descriptions

The experimental dataset[4] comes from AI competition held by Ant Financial Service Group. This competition aims at seeking better question similarity calculation method for customer service. The dataset contains 102477 question pairs with manual labeled and we can take this label as experimental ground truth. The average length of questions is 13.4. During the data preprocessing, we pad each question of same length (20 in word-level and 48 in character-level).

4.2 Evaluation Metrics

The parameter size and Macro F1-score are utilized as the evaluation metrics from two different aspects. The parameter size is the amount of model parameter and used to measure the complexity and time cost of different models, and Macro F1-score aims to evaluate the performance of different models in measuring sentence similarity.

4.3 Training Details

We divide the dataset into development set and test set. The test set has 10000 question pairs to be predicted. We further divide the development set into 10-fold to perform training and use these 10 models for testing. The final results are the average of 10 models.

The optimization method we choose for model training is the Adam[12]. The first momentum is set to be 0.9 and the second 0.999. The initial learning rate is 0.001 and the batch size is 128. The hidden states of GRU have 192 dimensions and linear layers have 384 dimensions. The two CNN layers both have 128 filters with 3 different sizes 1, 2, 3 and word embeddings have 100 dimensions. We set dropout with a rate of 0.5, which is applied to all feedforward connections.

We initialize word and character embeddings with the 100-dimensional pre-trained cw2vec[4] vectors from the competition dataset. For the out-of-vocabulary (OOV) words, we initialize the embeddings randomly. All vectors including word and character embeddings are updated during training.

The entire model is trained end-to-end and take cross-entropy as loss function. For all the experiments, we pick the model which works the best on the development set, and then evaluate it on the test set.

[4] `https://dc.cloud.alipay.com/index#/topic/data?id=3`

4.4 Experimental Results

In this subsection, we compare our model with state-of-the-art models on the paraphrase identification task. Because Ant Financial competition dataset is a brand-new dataset and no previous results have been published yet, we implement the following baseline models to compare:

Table 1. Experimental results and corresponding parameter sizes on Ant Financial dataset.

Model	Paras	Train	Test
(1) Siamese-CNN[24]	994K	0.5223	0.6021
(2) Siamese-RNN[17]	1847K	0.5462	0.6235
(3) Siamese-RCNN[13]	1674K	0.5668	0.6499
(4) Siamese-Attention-RCNN[8]	1723K	0.5725	0.6526
(5) ABCNN[23]	1536K	0.560	0.6346
(6) ESIM[5]	2118K	0.5870	0.6915
(7) ERCNN	**2077K**	**0.6022**	**0.7014**

– Siamese-CNN[24]: utilizes CNN to encode input sentences into vectors and measures similarity by the cosine distance.
– Siamese-RNN[17]: utilizes BiLSTM to encode input sentences into vectors and measures similarity as same as Siamese-CNN.
– Siamese-RCNN[13]: utilizes RCNN to encode input sentences into vectors and measures similarity by concatenating sentences vecotors and performing logistic regression.
– Siamese-Attention-RCNN[8]: utilizes attention-based RCNN to encode input into sentences vectors and measures similarity as same as Siamese-RCNN.
– ABCNN[23]: utilizes attentionbased CNN to encode input sentences and capture interactive features and measures similarity as same as Siamese-RCNN.
– ESIM[5]: utilizes matching-aggregation framework to encode input sentences and perform multiperspective matching interactions to measure similarity.

We are concerned with both parameter size and Macro F1-score when evaluating different models. We list the parameter sizes when they reached the best performances on test dataset, and table 1 shows the detailed comparisons.

The experimental results display that our ERCNN has outperformed all the previous models in both development dataset and test dataset. From the results of model (1), (2) and (3), we can see that Siamese-RNN performs better than Siamese-CNN in paraphrase identification task. But with the enhancement of CNN, Siamese-RCNN achieves better results than Siamese-RNN. This finding implies us that the incorporation of both CNN and RNN structures can deepen learning the sentence similarity as CNN is better to extract the robust and abstract features while RNN performs better to capture the fine granularity features. These parts of experimental results also demonstrate that our improvements on ESIM are reasonable and effective. On the other hand, in view of the comparison of parameter size, we show that the model with the combination of CNN have better performance with lower complexity. For example, Siamese-RCNN has less parameters than Siamese-RNN but achieves higher Macro F1-score in experiments.

And from the results of model (4) and (5), we can discover that with the addition of attention layer, the performance of each model performs better indeed as attention layer help extract the mutual information and better measure similarity between two sentences in comparation with other models which neglect these type of information.

Table 2. Partial prediction results of ESIM and ERCNN.

Question Pairs	Label	ESIM	ERCNN
花呗是否需要绑定银行卡 Does Huabei need to be bound with credit card? **花呗可不可以绑定银行卡** Can Huabei be bound with credit card?	1	✓	✓
花呗怎么还款 How to repay Huabei? **花呗可不可以绑定银行卡** How do I repay my Huabei?	1	×	✓
花呗绑定另一个手机号能解绑吗 Can Huabei be unbound with other mobile phone number? **我解绑花被让自己的另一个手机号使用花呗** Can I unbound Huabei to use other phone number?	1	×	✓
怎么保证花呗安全 How to ensure the security of Huabei? **安全验证没有成功花呗** Security certification doesn't succeed Huabei.	0	×	✓
花呗显示负的是什么意思 What is the meaning of the Huabei showing negative? **花呗显示负***是总共还欠着***吗** Huabei shows negative *** means that owed a total of ***?	1	×	×

At the end of the Table 1, the comparison results of model (6) and (7) shows that our improvements based on ESIM have achieved satisfied results, from aspects of both evaluation metric and model complexity. The most difference between our model and ESIM lies in the supplement of CNN layer after original input encoding. As our former experimental results shown, CNN can help extract more elaborated feature in sentences and reduce the model complexity. Moreover, compared with ESIM, we apply feedforward layer to replace the origin BiLSTM layer in overall similarity modeling, and this modification of the model assists us further in reducing the model parameter size.

4.5 Further Analysis with Typical Cases

In this subsection, we analyze partial typical prediction results of ESIM and our ERCNN. Table 2 shows these partial prediction results. By comparing the prediction results for different type of question pairs, we can show that our ERCNN performs better than ESIM in learning similarity between two sentences even if two sentences don't have much common words or they have some wrongly written or mispronounced characters.

From the first case in table 2, we can see that for two semantically similar sentences with much words or characters in common, ESIM and ERCNN are both able to measure the similarity despite the sentences lengths are short.

And for the second case, we choose a question pair with some misspellings ("花被" is wrongly written and should be "花呗"). In such condition, the prediction result of ESIM is wrong due to the reason that it only makes use of BiLSTM to consider the sequence information and ignore the key word information, but our ERCNN with CNN as a supplement can better identify the relation between "花被" and "花呗", and therefore is not easily to be mislead by misspellings.

Then, for the third and fourth cases, the question pairs to be compared both don't look like grammatical sentences. Because in the scene of intelligent customer service, people post their questions in mobile devices and often full of grammatical mistakes, so a practical model should be able to handle this condition and identify the client's real purpose. The third and fourth cases show comparisons between ESIM and ERCNN in above-mentioned condition. As the result turns out, ESIM is mislead by the semantic confusion and can't identify the similar question pair, but our ERCNN makes out in this condition because the incorporation of CNN improves the ability to capture robust and abstract features. With such enhancement, our model is relatively hard to be puzzled by grammatical mistake in sentences.

In the final case, we're supposed to infer that ESIM and ERCNN both misjudge the similarity of two sentences. We believe the reason is that the two models are mislead by the specific symbol such as "*" in the sentences. With the development of mobile internet, an increasing number of specific symbols or emoticons appear in our daily network communication[9]. These specific symbols are meaningful and full of sentiment information[2]. Our ERCNN and ESIM both don't take these specific symbols into consideration when measuring similarity and we need to attach suffiecient weight to them in future work.

5 Conclusions and Future Work

In this paper, we have introduced a new model named ERCNN based on ESIM for paraphrase identification task. ERCNN achieves the state-of-the-art performance in learning sentence similarity on Ant Financial competition dataset. Our model demonstrates the incorporation of CNN in local similarity modeling which helps extract more elaborated information and puts the fullyconnected layers to replace the time-consuming BiLSTM layers in overall similarity modeling with lower complexity. In the future work, we plan to extend the ERCNN model to other NLP tasks like question answering and natural language inference, and we will pay more attention to the specific symbols and emoticons in sentences to adapt to the reality.

Acknowledgments

We thank the anonymous reviewers for their helpful comments on this paper. This work was partially supported by National Natural Science Foundation of China (61572049 and 61876009).

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Barbieri, F., Ronzano, F., Saggion, H.: What does this emoji mean? a vector space skip-gram model for twitter emojis. In: LREC (2016)
3. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a" siamese" time delay neural network. In: Advances in neural information processing systems. pp. 737–744 (1994)
4. Cao, S., Lu, W., Zhou, J., Li, X.: cw2vec: Learning chinese word embeddings with stroke n-gram information. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
5. Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H., Inkpen, D.: Enhanced lstm for natural language inference. arXiv preprint arXiv:1609.06038 (2016)
6. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
8. Du, C., Huang, L.: Text classification research with attention-based recurrent neural networks. International Journal of Computers Communications & Control **13**(1), 50–61 (2018)
9. Eisner, B., Rocktäschel, T., Augenstein, I., Bošnjak, M., Riedel, S.: emoji2vec: Learning emoji representations from their description. arXiv preprint arXiv:1609.08359 (2016)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

13. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: Twenty-ninth AAAI conference on artificial intelligence (2015)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
15. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
17. Mueller, J., Thyagarajan, A.: Siamese recurrent architectures for learning sentence similarity. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
18. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). pp. 807–814 (2010)
19. Neculoiu, P., Versteegh, M., Rotaru, M.: Learning text similarity with siamese recurrent networks. In: Proceedings of the 1st Workshop on Representation Learning for NLP. pp. 148–157 (2016)
20. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
21. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. pp. 3104–3112 (2014)
22. Wang, Z., Hamza, W., Florian, R.: Bilateral multi-perspective matching for natural language sentences. arXiv preprint arXiv:1702.03814 (2017)
23. Yin, W., Schütze, H., Xiang, B., Zhou, B.: Abcnn: Attention-based convolutional neural network for modeling sentence pairs. Transactions of the Association for Computational Linguistics **4**, 259–272 (2016)
24. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in neural information processing systems. pp. 649–657 (2015)