BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 05/06 pp234–244
Volume 1, Number 3, September 2018
DOI: 10.26599/BDMA.2018.9020022

Relation Classification via Recurrent Neural Network with Attention and Tensor Layers

Runyan Zhang, Fanrong Meng*, Yong Zhou, and Bing Liu

Abstract: Relation classification is a crucial component in many Natural Language Processing (NLP) systems. In this paper, we propose a novel bidirectional recurrent neural network architecture (using Long Short-Term Memory, LSTM, cells) for relation classification, with an attention layer for organizing the context information on the word level and a tensor layer for detecting complex connections between two entities. The above two feature extraction operations are based on the LSTM networks and use their outputs. Our model allows end-to-end learning from the raw sentences in the dataset, without trimming or reconstructing them. Experiments on the SemEval-2010 Task 8 dataset show that our model outperforms most state-of-the-art methods.

Key words: semantic relation classification; bidirectional Recurrent Neural Network (RNNs); attention mechanism; neural tensor networks

1 Introduction

A relation classification task identifies the semantic relationship between two marked entities within a text. For example, considering the context in the given sentence, "The ⟨e1⟩news⟨/e1⟩ brought about a ⟨e2⟩commotion⟨/e2⟩ in the office.", there is a relationship between the two marked entity mentions, e1 ("news") and e2 ("commotion"), and the sentence can be expressed as follows: the "news" causes a "commotion". This is denoted with label "Cause-Effect".

Relation classification is an important component in Natural Language Processing (NLP) systems, such as question answering and information extraction. Accurate relationship classification facilitates precise understanding of semantic meaning and promotes targeted analysis of sentences. Thus, relation classification has attracted substantial attention in recent years.

Many traditional models focus on machine learning and feature design, which usually rely on a full-fledged NLP pipeline and require extra hand-crafted features or kernels[1]. In recent years, deep learning methods have been widely used for relation classification and provide an effective method for reducing artificial adjustments. Several neural network models have been proposed for this task, such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and other neural architectures. Some are built with the shortest dependency paths and dependency subtrees[2–4], whereas others input the raw sentences with few or no pre-trained operations to learn implied features[5–7]. These approaches have been proven effective but would be inevitably disturbed by irrelevant parts.

Thus, some attention-based approaches have been proposed to make the model automatically focus on the most useful part. For such approaches, it is sufficient to use raw sentences as input[8–10]. The attention mechanism performs well in considering the contribution of each part but performs unsatisfactorily when contexts are insignificant. For example, for the sentence, "This 8-day ⟨e1⟩music⟨/e1⟩ ⟨e2⟩clock⟨/e2⟩

• Runyan Zhang, Fanrong Meng, Yong Zhou, and Bing Liu are with China University of Mining and Technology, Xuzhou 210009, China. E-mail: zhangrunyan@cumt.edu.cn; mengfr@cumt.edu.cn; yzhou@cumt.edu.cn; liubing@cumt.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2018-02-03; accepted: 2018-03-01

needs winding only once a week," it is difficult to detect the relationship between "music" and "clock" by context. however, it is easier when the meaning of "music clock" is known.

Features of entities are as important as those of context and are usually complex. Hence, inspired by the Recursive Neural Tensor Network (RNTN), which was proposed by Socher et al.[11], we use a tensor layer to extract the hidden features between entities. It is "a more direct, possibly multiplicative, interaction allowing the model to have greater interactions between the input vectors"[12]. The key contributions of our approach are as follows:

(1) Our Bidirectional RNN (which uses Long Short-Term Memory, LSTM, cells) relies on a word-level attention mechanism. This allows the model to make full use of the context, thereby enabling it to learn which parts are effective for a given classification task.

(2) We use a tensor layer to discover additional interactions between entities. Then, we consider both attention features and tensor features for classification.

(3) We evaluate our model on the SemEval 2010 Task 8 dataset and achieve a state-of-the-art result with an F1-score of 86.3

2 Related Work

Semi-supervised and unsupervised methods for relation classification have been proven effective and adaptable in the field of networks[1,2]. However, in this paper, we mainly consider supervised methods, which usually have higher accuracy.

In earlier studies of relationship classification, researchers focused on carefully selecting features through a series of NLP tools[1] or elaborately designing kernels. In addition, the Distant Supervision method was proposed for problems without a training corpus, which aligns the knowledgebase with the unstructured text. SemEval-2010 task 8 was published as a standard relation classification corpus[13], and many follow-up studies have utilized it.

Traditional methods have been proven to be effective, but they strongly depend on the quality of the designed model. With the recent rise of interest in DNN, many neural-network-based models have been used to extract hidden features. The Recursive Matrix-Vector Model by Socher et al.[14] adds a matrix to each node in the RNN to enhance the fitting ability of the entire network. Yu et al.[15] proposed Factor-based Compositional embedding Models (FCM), which decompose the sentence into substructures, extract features independently, and combine them via a sum-pooling layer. Xu et al.[16] proposed a Deep Recurrent Neural Network (DRNN) model, which splits the sentence into two parts by the root word of the parse tree and inputs the parts into a multi-layer RNN, in which the neurons consider the information of adjacent words in the previous layer. In the works described above, the models are designed according to the parse tree and achieve good performance. In contrast, some researchers construct end-to-end models, which use raw sentences as the input of neural networks and embed NLP features to describe words. Zeng et al.[7] proposed a CNN for extracting lexical and sentence-level features. Vu et al.[5] proposed extended middle context for CNNs and combined it with bidirectional RNNs using a simple voting scheme. The end-to-end model is concise and easy to implement, but unlike parse-tree-based models, it uses the shortest dependency subtree; thus, it may be disturbed by meaningless words.

Over the past three years, some attention-based models have been proposed, which can automatically adjust the weights of each word. In NLP, the attention mechanism was first used by Bahdanau et al.[17] for text alignment in machine translation. It quickly attracted attention and was widely used for many other tasks. Zhou et al.[8] combined the attention mechanism with bidirectional RNNs. On this basis, Xiao and Liu[9] split the sentence into two entities and used two attention-based RNNs hierarchically. Wang et al.[10] proposed a novel CNN that relies on input-level attention and pooling-level attention. The inputs of these attention-based methods are all raw sentences, with few embedded NLP features, and they achieve good performance.

In addition to the attention mechanism, this paper addresses the tensor layer. Like the Neural Tensor Network (NTN)[12], the tensor layer trains a high-order tensor as a weight between multiplying inputs and can use bilinear interaction instead of a traditional linear combination. Thus, it has a stronger ability to express the relationships among the inputs. Qiu and Huang[18] proposed a convolutional neural tensor network for community-based question answering and used a tensor layer to model the relations between the question and its answer. Pei et al.[19] proposed a Max-Margin Tensor Neural Network (MMTNN), whereby a tensor

layer was used to extract the hidden combinations of features. It is obvious that the tensor layer is effective in modeling relations between the inputs.

3 Model

In this section, we propose our model in detail, termed the Attention and Tensor based Recurrent Neural Network (AT-RNN). As shown in Fig. 1, the model first extracts the hidden features of the sentence through a two-level LSTM network, of which the first level is a bidirectional LSTM block and the second level is a simple LSTM block. Then, the advanced features are further extracted by two operations: the word-level attention and the entity-level tensor layer. Finally, the two kinds of vectors are combined for semantic relation classification:

- Word-level attention: The model calculates a weight for every word with respect to the last output of the LSTM networks and merges the features from each time step by multiplying them by the weight vector.
- Entity tensor layer: The model trains a tensor as a weight to extract additional interactions between entities. More details can be found in Section 3.4.

3.1 Input representation

Word embeddings, positions, and parts of speech are the three kinds of features used to transform each word into a real-valued vector. We also perform a comparison experiment with only word embeddings and achieve a slightly worse performance, with an F1-score of 85.2.

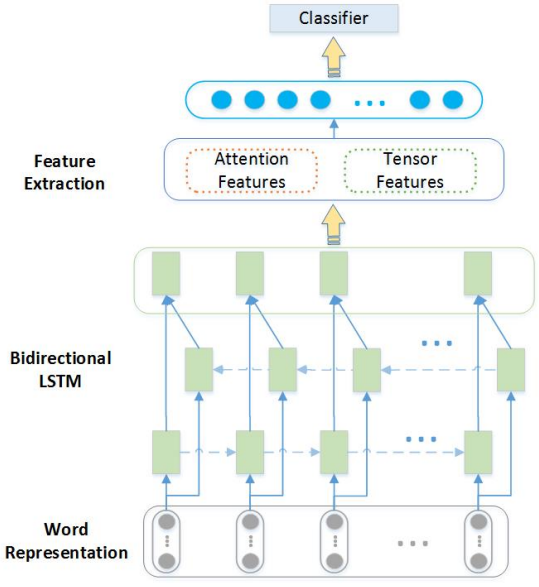


Fig. 1 The architecture of our model.

Given a sentence, $S = (w_1, w_2, ..., w_n)$, we convert each word into a semantic vector by looking up the word embedding matrix $W_{\text{word}} \in \mathbb{R}^{|V| \times d_w}$, where d_w is the dimension of the semantic vector and $|V|$ is the size of the training vocabulary of the word embedding matrix. After looking up the matrix, every word is mapped to a row vector $w_i^d \in \mathbb{R}^{d_w}$.

For the polysemy problem, we use the Part Of Speech (POS) tag to enhance the semantic representation of words. By looking up the POS embedding matrix, we map each POS tag to a pre-trained vector $w_i^{\text{POS}} \in \mathbb{R}^{d_{\text{POS}}}$, where d_{POS} is a hyper-parameter.

We also use position embedding to describe the relative distances between two entity mentions and each word[7, 10]. The two entity mentions are calculated separately. For example, in the sentence "That ⟨e1⟩machine⟨/e1⟩ makes a lot of ⟨e2⟩noise⟨/e2⟩", the relative distances of the word "makes" to entity "machine" and entity "noise" are 1 and −4. Thus, we will obtain two vectors $w_{i1}^p, w_{i2}^p \in \mathbb{R}^{d_p}$ for the position feature, where d_p is a hyper-parameter.

Finally, we combine the three kinds of features for each word to construct the input $w_i^f = [w_i^d, w_i^{\text{POS}}, w_{i1}^p, w_{i2}^p]$.

3.2 Multi-level rnns

RNNs have been widely used in many NLP tasks. They can extract hidden features by considering the entire sequence. However, for deep iterations, RNNs may suffer from the problem of exploding or vanishing gradient. In other words, input for which the distance is long may provide inappropriate feedback. To solve this problem, LSTM units were proposed by Hochreiter and Schmidhuber[20]. The main idea is to use linear memory cells to store information of each iteration and to utilize three gate units for reading, writing, and resetting. Many LSTM variants have been designed for specific tasks. In this paper, we adopt the LSTM variant that was proposed by Zaremba et al.[21]

The LSTM cell is comprised of a memory unit c_t, which is analogous to a peephole and allows information to flow through the cell without interaction. To change the information in the memory unit, the LSTM cell sets three logical gates: an input gate i_t for writing, a forget gate for removing and an output gate for reading. Each gate provides the percentage of information that is involved in the corresponding operation, which is determined by the input x_t and the previous hidden state h_{t-1}. The detailed calculations

are shown in Eqs. (1)–(6):

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (2)$$

$$g_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (3)$$

$$c_t = i_t g_t + f_t c_{t-1} \quad (4)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (5)$$

$$h_t = o_t \tanh(c_t) \quad (6)$$

Generally, the semantic meaning of a word will be affected by both the previous context and the subsequent context. Hence, we use bidirectional LSTM networks, which consist of two LSTM blocks with forward and backward sequence inputs. We place another simple LSTM block after the bidirectional LSTM blocks, which can organize previous features on the word level, and the last output will contain semantic information of the entire sentence. In the following sections, all RNNs that are mentioned in our model refer to the LSTM variant.

3.2.1 Word-level attention

Since each word may make a different contribution to the classification, we introduce a novel word-level attention mechanism, which is driven by RNN features of each word and the entire sentence. The attention mechanism will concentrate more on the words that are important for prediction and give them greater weights. The working structure of the sentence attention layer is shown in Fig. 2. The attention weight that corresponds to each word is calculated via a non-linear combination of the hidden state h_t and the context vector h_{last}. Through the weighted sum of the word states, we obtain the attention features of the sentence.

Motivated by Luong et al.[22], we consider three different scoring functions for the combination of attention weight parameters h_t and h_{last}. We only need one output, so there is only one representation a in the equations, as shown in Eqs. (7)–(10):

$$a = V_{atn}^{\mathrm{T}} \sum_{t=1}^{n} a_t h_t \quad (7)$$

$$\alpha_t = \frac{\exp(e_t)}{\sum_{j=1}^{n} \exp(e_j)} \quad (8)$$

$$e_t = v_a^{\mathrm{T}} \text{score}(h_t, h_{\text{last}}) \quad (9)$$

$$\text{score}(h_t, h_{\text{last}}) = \begin{cases} h_t^{\mathrm{T}} h_{\text{last}}, & \text{DOT;} \\ h_t^{\mathrm{T}} W_a h_{\text{last}}, & \text{GENERAL;} \\ \tanh(W_a[h_t : h_{\text{last}}]), & \text{CONCAT} \end{cases} \quad (10)$$

where a is the attention representation of the sentence, $V_{atn} \in \mathbb{R}^{d_h \times d_a}$ is a transform matrix for a, and α_t is a real number that indicates the weight rate. To adjust the proportion between attention features and tensor features, we set the output dimension d_a as a hyper-parameter.

3.2.2 Entity tensor layer

Tensor networks can describe a larger number of interactions between inputs and model deeper relations. In this paper, we place a tensor layer after the RNNs for two entities. As shown in Fig. 3, given the hidden features of the RNNs that correspond to two entities h_{e1} and h_{e2}, we combine them in succession and extract features through a bilinear form. For this target, we train two weights: one with three dimensions for the square interaction and the other with two dimensions for linear interaction. Equations (11)–(13) define the output t of the tensor layer:

$$t = f(p_b + p_l) \quad (11)$$

$$p_b = [h_{e1} : h_{e2}]^{\mathrm{T}} W_t 1[h_{e1}, h_{e2}] \quad (12)$$

$$p_l = W_{t2}[h_{e1}, h_{e2}] \quad (13)$$

The dimension of the weight matrix W_{t1} is $d_t \times 2d_h \times 2d_h$, where d_h is the hidden size of the RNNs and d_t is the output dimension of the tensor layer (which is a hyper-parameter).

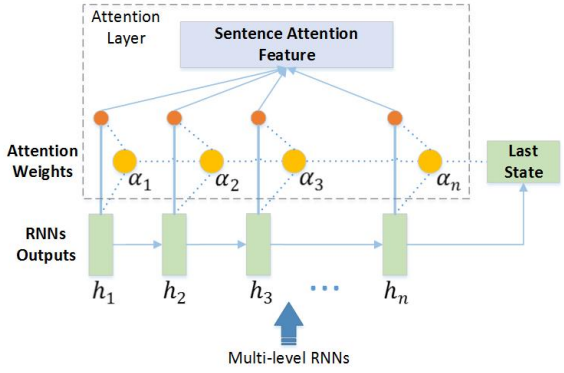


Fig. 2 Architecture of the attention layer.

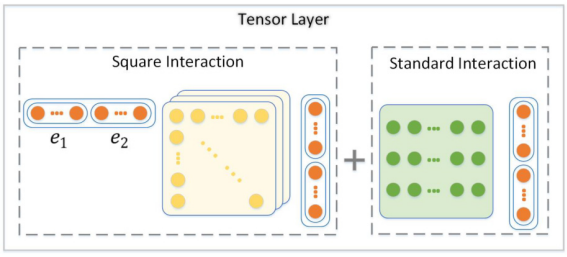


Fig. 3 Architecture of the tensor layer.

If we decompose the matrices in the equation for p_b, we will find the mathematical principles of the tensor layer. Given the input as $[x_1, x_2, ..., x_m]$ and a slice of the weight matrix

$$\begin{bmatrix} w_1 & \cdots & w_{m^2-m+1} \\ \vdots & \ddots & \vdots \\ w_m & \cdots & w_{m^2} \end{bmatrix},$$

the square interaction can be calculated as Eq. (14):

$$p_b^i = w_1 x_1 x_1 + w_2 x_1 x_2 + \cdots + w_m x_1 x_m + \cdots + w_{m^2} x_m x_m \quad (14)$$

or as Eq. (15):

$$p_b^i = \sum_{j=1,\, k=1}^{m} w_{j\times k} x_j x_k \quad (15)$$

We observe that in the bilinear part of the tensor layer, the model tries every second-order combination of inputs. Hence, the tensor layer can relate the two inputs multiplicatively instead of only implicitly through the nonlinearity, as with standard neural networks, where the entity vectors are simply concatenated[12].

Furthermore, the tensor layer can utilize a special attention mechanism on a single input. Here, we present the analysis and derive the equation. With the input $[x_1, x_2, ..., x_m]$, the attention-weighted integral of the input can be calculated as Eq. (16):

$$t^i = \sum_{j=1}^{m} x_j \ \text{score}([x_1 : x_m]) \quad (16)$$

where the scoring function is same as Eq. (9); however, here we define it using a linear evaluation:

$$\text{score}([x_1 : x_m]) = \sum_{k=1}^{m} w_k x_k + b \quad (17)$$

We substitute Eq. (17) into Eq. (16):

$$t^i = \sum_{j=1,k=1}^{m} w_{jk} x_j x_k + \sum_{j=1}^{m} b_j x_j \quad (18)$$

where the left part is equivalent to the square interaction p_b of the tensor layer and the right part is equivalent to the standard interaction p_l of the tensor layer.

The weight matrix t_1 is $M \times N \times N$-dimensional. Therefore, $O(MN^2)$ computational resources are required to calculate it exactly, where M is the size of the tensor layer and N is the input size. However, the dimensions of the word vectors are usually in the range of 50 to 300. Thus, it will not consume much more time compared with a single-layer network with complexity $O(MN)$. In addition, the multiplication of the matrix is highly parallelizable and the calculation of the square interaction can be completed in two multiplications.

3.2.3 Regularization and training

Dropout method is to reduce the complexity of the network by obstructing a proportion of the cells in the neural networks. Zaremba et al.[21] improved LSTM networks so that dropout could be correctly applied. In this paper, we adopt a dropout wrapper for the LSTM cells and an output layer with rates of 0.6 and 0.4. In addition, we constrain the L2-norms of the weight vectors. The implementation of the back-propagation algorithm with L2-regularization can be defined as Eq. (19):

$$\theta \leftarrow \theta + \lambda \frac{\partial J(\theta)}{\partial \theta} + \lambda_2 \frac{\partial \beta ||\theta||^2}{\partial \theta} \quad (19)$$

where λ and λ_2 are learning rates and $J(\theta)$ is the loss function, for which we use the cross-entropy function, which is defined as Eq. (20):

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} y^i \log P(y^i|x^i, \theta) + (1 - y^i) \log(1 - P(y^i|x^i, \theta)) \quad (20)$$

4 Experiment

4.1 Dataset and experimental setup

Dataset: We evaluate our model on the commonly used dataset SemEval-2010 Task 8[13], which contains nine types of relations and an "Other" type for the entity relations that are not of any of these nine types. The nine types are Cause-Effect, Instrument-Agency, Product-Producer, Content-Container, Entity-Origin, Entity-Destination, Component-Whole, Member-Collection, and Message-Topic. Considering the distinction between active and passive in relationships, the dataset annotates the direction for the nine relation types, while the "Other" type is not directional, so the total number of relation types is 19. The SemEval-2010 Task 8 dataset consists of 8000 sample training sets and 2717 sample test sets. We use the official scorer, using macro-averaged F1-score (excluding Other), to evaluate the model performance.

Settings: We use the word2vec skip-gram model[23], which is trained on Wikipedia, to embed words into vector representations. We also try the Senna model and the glove model for word embedding. We pretrain d_{POS}-dimensional embeddings for POS features and d_p-dimensional embeddings for position features. To automatically identify the POSs of the words, we use the Stanford POS Tagger toolkit. We initialize the weight matrices with random values using Gaussian

distributions. We use the Adam optimization method to update matrices iteratively. Some detailed settings of hyper-parameters are listed in Table 1.

4.2 Experimental results and comparison

4.2.1 Comparison with previous works

Table 2 provides a comparison of F1-scores among our model and the previous works. We divide them into four categories, and we observe that our model's performance is comparable to those of the state-of-the-art models.

Non-Neural Networks Models. We chose a representative method with an SVM classifier[24], which considers eight sets of original features: Lexical, Dependency, PropBank, FrameNet, Hypernym, NomLex-Plus, NGrams, and TextRunner. In the paper on the SVM, the researchers tried different combinations of the eight features and found that the features all contribute to classification. This method was the winner of the SemEval task at the time, with an F1-score of 82.19, but was later outperformed. This is because recent research focuses on the in-depth exploration of the original information, mainly through the neural networks, instead of using a well-designed classifier for the carefully selected original features, which may not fit well and may suffer extra deviations. An increasing number of studies have shown that a small number of the NLP features is sufficient for classification through hidden feature extraction.

Table 1 Hyperparameters.

Parameter	Parameter name	Value
d_w	Word Emb. Size	100
d_{POS}	POS Emb. Size	25
d_p	Position Emb. Size	25
d_h	RNNs Hidden Size	200
$d_a + d_t$	Feature Extraction Size	120
λ	Learning Rate	0.01
λ_2	L2 Learning Rate	0.0001

Table 2 Comparison with previous results on SemEval 2010 task 8 dataset.

Classifier	F1-score
Non-Neural Networks Model	
SVM[24]	82.2
Shortest Dependency Tree (SDP) Model	
MVRNN[14]	82.4
FCM[15]	83.0
DepNN[3]	83.6
depLCNN+NS[2]	85.6
SDP-LSTM[4]	83.7
DRNNs[16]	86.1
End-To-End Model	
CNN[7]	82.7
BLSTM[6]	84.3
ER-CNN + R-RNN[5]	84.2
Attention-based Model	
Hier-BLSTM[9]	84.3
Att-BLSTM[8]	84.0
Attention-CNN[25]	85.9
Multi-Att-CNN[10]	88.0*
Our Model	
AT-BLSTM (word2vec) (word dim=100)	86.3

Notes: We failed to reproduce the positive result with the Multi-Att-CNN model and the performance of our implementation is about 85.5. We also found an experiment for the paper on the GitHub https://github.com/FrankWork/acnn, which failed to re-implement the result as well, though it may need more optimizations. We think the problem is likely due to insufficient details in the paper for the pooling attention and some tricks for the use of relation labels.

SDP Models. SDP is a reasonable method for detecting semantic structure and logic. It eliminates the irrelevant words to the relationship between two entity nominals and constructs a tree framework for the remaining words, according to which the parent node has a direct effect on the child node. Hence, the models that are based on the SDP can ignore the meaningless words and the input itself will be structured with connections. However, this is the ideal case. In practice, the SDP may not always be accurate and the parsing time will increase exponentially for long sentences. Nevertheless, from the experimental results, the models that are based on SDP have obvious advantages. Moreover, in other research, the attention mechanism is used to recognize relevant words automatically, which can replace the SDP pretraining and avoid the extra incorrect information from the parse tools.

End-To-End Models. With the development of deep learning, some researchers have pinned their hopes on making networks extract features automatically. Instead of concentrating on selecting the semantic presentation or structure, they reform the networks to make them more powerful and robust. According to the experimental results, these models have advantages over previous works and require very little artificial participation while pretraining. However, when the input is an entire sentence, the end-to-end models inevitably suffer from uncorrelated noise disturbances.

Attention-based Models. The attention mechanism provides a weight for each part of the input, which reduces the interference of the noise words. It assigns larger weights to significant words to reinforce their

impact on the classification prediction. As shown Table 2, the attention-based models outperform the end-to-end models due to the special treatment of the noise words. As in SDP-based models, instead of removing the irrelevant words from the sentence, the attention-based models assign small weights to the irrelevant words (which may not be completely ineffective). In addition to the attention mechanism, our model uses a tensor layer to extract complex connections between two entities, which play essential roles when the context is not particularly helpful. Therefore, our model performs better than most attention-based models.

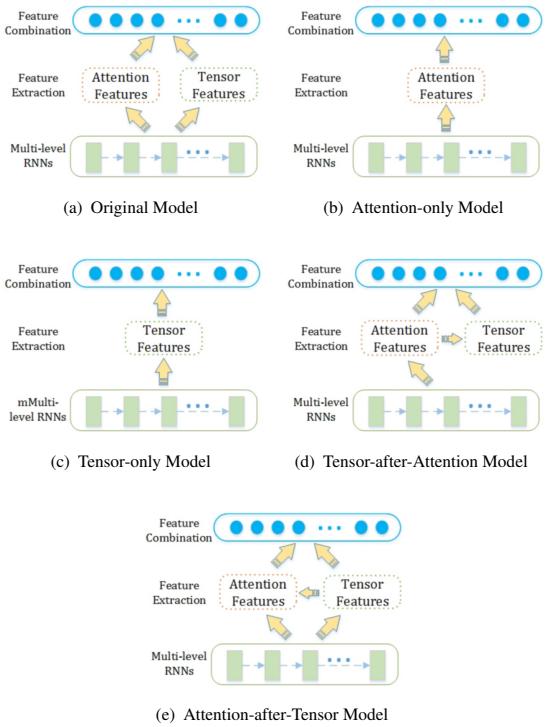
4.2.2 Variants of the model

To evaluate effects of the attention layer and the tensor layer, we construct four variants of the model, as shown in Fig. 4.

Original model in Fig. 4a. In the original model, advanced features are extracted. The inputs of both the attention layer and the tensor layer are the hidden features that are outputted by the RNNs.

Attention-only model in Fig. 4b. In this variant, we only use an attention layer to extract features after the RNNs. It is similar to Att-BLSTM[8], but we place an additional basic LSTM block after the BLSTM block.

Tensor-only model in Fig. 4c. In this variant, only a tensor layer is used after the RNNs. Comparing with the attention-only model, we observe that the tensor layer performs better. This is because the entity inputs carry the context information through the multi-RNNs, although not as well as the attention layer. Then, the interactions of the relations are extracted through the tensor layer.



(a) Original Model (b) Attention-only Model

(c) Tensor-only Model (d) Tensor-after-Attention Model

(e) Attention-after-Tensor Model

Fig. 4 Variants of the model.

Tensor-after-attention model in Fig. 4d. In this variant, we replace the input of the tensor layer by the output of the attention layer, as Eq. (21):

$$t = f(a^{\mathrm{T}} W_{t1} a + W_{t2} a) \tag{21}$$

where a is the output of the attention layer. We find that the tensor-after-attention model is less effective than the original model. We conjecture that this is because the attention layer combines the entire sentence information and lacks the pertinence of the entity pairs. Thus, the relation features between entities are not strengthened much by the tensor layer.

Attention-after-tensor model in Fig. 4e. Instead of calculating the attention weights by the last output of the RNNs, we consider the output of the tensor layer, as Eq. (22):

$$\alpha_i = \mathrm{softmax}(v_a^{\mathrm{T}} \mathrm{score}(h_i, t)) \tag{22}$$

where α_i is the attention weight of the i-th state of the RNNs and t is the output of the tensor layer. We observe that the effect of the attention-after-tensor model is roughly equivalent to that of the original model, with higher recall rate but lower accuracy rate. Weighted sentence combination using the attention layer, which is driven by the tensor layer, is available and effective. However, it may require longer training time than the original model.

4.3 Detailed analysis

4.3.1 Attention scoring functions

In this section, we compare different scoring functions, as shown in Eq. (10), for the calculation of the attention weights. For convenience, we use the attention-only variant as shown in Fig. 4b. Figure 5 shows the training losses of three models and Table 3 gives the F1-scores.

We observe that the DOT function does not well fit our model and requires more time to train, since it is inadvisable to evaluate the attention weight using direct multiplication. The GENERAL function and the CONCAT function give similar performances and convergence rates, but the loss of the CONCAT function declines more smoothly due to its simpler representation. In our model, we adopt the CONCAT function to calculate the attention weights.

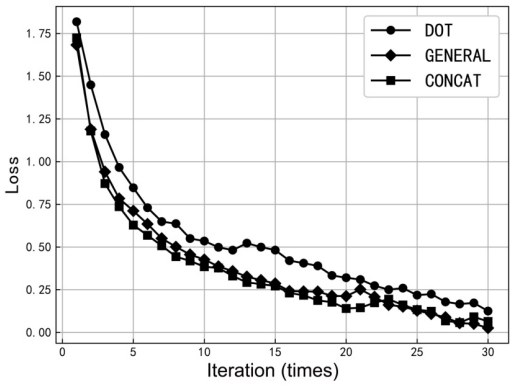


Fig. 5 Loss variation of the attention scoring functions.

Table 3 Results of the scoring functions.

Scoring function	F1-score
DOT	77.4
GENERAL	84.2
CONCAT	84.3

4.3.2 Attention scoring functions

Figure 6 shows the distribution of word-level attention weights, where Fig. 6a is a positive case and Fig. 6b is a negative case. For the sentence "The most common ⟨e1⟩audits⟨/e1⟩ were about ⟨e2⟩waste⟨/e2⟩ and recycling", the relation between the two entities is "Message-Topic", and the model assigns the greatest weight to the word "about" and minor weights to the words "the most common audits", which is semantically reasonable. Thanks to auto-weighting by the attention layer, the model will identify the most highly related words and will be easier to classify.

However, for the sentence "My ⟨e1⟩shoe⟨/e1⟩ ⟨e2⟩laces⟨/e2⟩ stay tied all the time" with the relation "Component-Whole", the context offers little help, so the effect of the attention mechanism is less than satisfactory. This is an extreme case, and it will be ameliorated with the appropriate word vector representation.

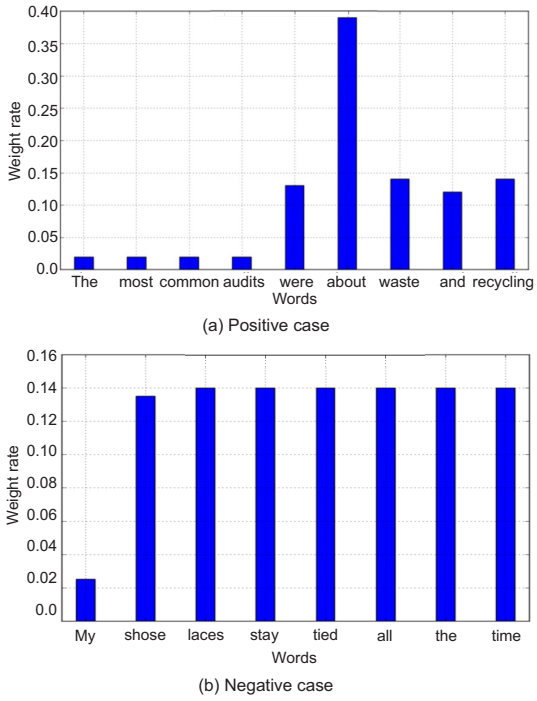


Fig. 6 Graphical representation of attention weights.

4.3.3 Effect of the tensor layer

Comparing the results of the attention-only model and those of the other models in Table 4, we observe that the tensor layer improves the performance of the model by approximately 2%. The tensor operation is an effective means for extracting deeper and more complex connection information. In addition, according to Fig. 7, the convergence rate of the tensor layer is slightly lower than that of the attention layer; however, it is acceptable.

From Eq. (16) to Eq. (18), we have derived the equivalence between the tensor operation and the attention mechanism on a single input. The tensor layer

Table 4 Comparison of the variants.

Variant	F1-score
Original	86.3
Attention-only	84.3
Tensor-only	85.3
Tensor-after-attention	85.0
Attention-after-tensor	86.2

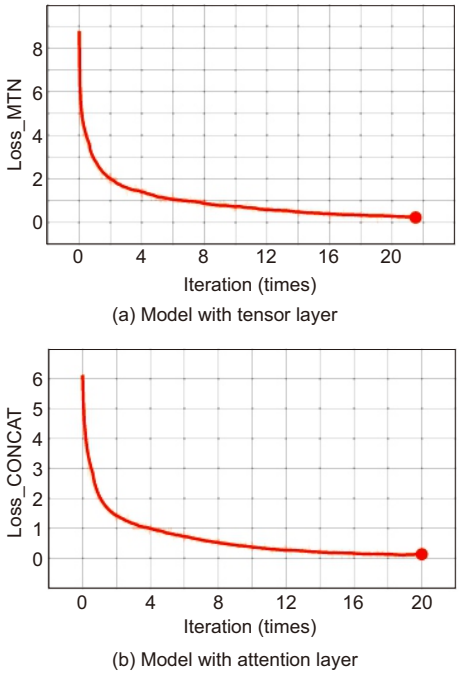


Fig. 7 Loss variances of tensor and attention.

can be viewed as an attention-weighted combination of all the dimensions in the input, which is driven by itself. One difference is that there is no softmax function for normalizing the attention weights, which can be considered multiple attentions that concentrate on multiple features.

4.3.4 Effect of the tensor layer

Figure 8 shows the F1-scores with different proportions of attention features and tensor features. We find that with 40 percent attention features and 60 percent tensor features, the model gives the best result. Furthermore, by analyzing the recall and accuracy rates, we observe the following two phenomena:

With the tensor layer, the model achieves higher recall rate but lower accuracy rate, which is ameliorated by increasing the percentage of attention features; the optimal performance is attained at the equilibrium point. This is because some hidden relations may be reasonable for the entity-level meanings, but are not indicated by the contexts, so the tensor layer gives an incorrect answer. However, with the introduction of the attention layer, the weight of the context information and the accuracy rate are increased.

The training speed increases when the attention layer is introduced. The tensor layer draws support from the multi-RNNs in extracting the context information, and the contexts are necessary. We need to cross the tensor layer to train the multi-RNNs, which will slow down the update speed. However, for the attention layer, we handle the contexts synchronously, and we can train the tensor layer and the attention layer at the same time. In summary, the tensor layer contributes greatly to relation classification and the attention layer improves the utilization of the contexts.

5 Conclusion

In this paper, we propose a novel bidirectional recurrent neural network architecture for the relation classification task, which utilizes an attention layer to organize the context information on the word level and a tensor layer to detect complex connections between two entities. Our model makes full use of the raw sentences in the dataset, without trimming or reconstructing them. In addition, we derive the connection between the attention and the tensor. Experiment on the SemEval-2010 Task 8 dataset shows that our model is effective and competitive. In future work, we intend to explore the application of the attention mechanism to recurrent neural tensor networks, and we will expand our model to other tasks.

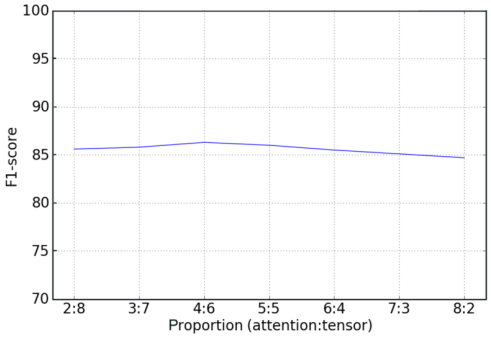


Fig. 8 Proportions of attention and tensor features.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61572505) and ChanXueYan Prospective Project of Jiangsu Province (No. BY2015023-05).

References

[1] F. M. Suchanek, G. Ifrim, and G. Weikum, Combining linguistic and statistical analysis to extract relations from web documents, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, USA, 2006, pp. 712–717.

[2] K. Xu, Y. S. Feng, S. F. Huang, and D. Y. Zhao, Semantic relation classification via convolutional neural networks with simple negative sampling, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[3] Y. Liu, F. Wei, S. J. Li, H. Ji, M. Zhou, and H. F. Wang, A dependency-based neural network for relation classification, in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Beijing, China, 2015.

[4] Y. Xu, L. L. Mou, G. Li, Y. C. Chen, H. Peng, and Z. Jin, Classifying relations via long short term memory networks along shortest dependency paths, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015, pp. 1785–1794.

[5] N. T. Vu, H. Adel, P. Gupta, and H. Schütze, Combining recurrent and convolutional neural networks for relation classification, in *Proceedings of NAACL-HLT*, San Diego, CA, USA, 2016.

[6] S. Zhang, D. Q. Zheng, X. C. Hu, and M. Yang, Bidirectional long short-term memory networks for relation classification, in *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, Shanghai, China, 2015, pp. 73–78.

[7] D. J. Zeng, K. Liu, S. W. Lai, G. Y. Zhou, and J. Zhao, Relation classification via convolutional deep neural network, in *Proceedings of COLING 2014, the 25th*

International Conference on Computational Linguistics: Technical Paper, Dublin, Ireland, 2014, pp. 2335–2344.

[8] P. Zhou, W. Shi, J. Tian, Z. Y. Qi, B. C. Li, H. W. Hao, and B. Xu, Attention-based bidirectional long short-term memory networks for relation classification, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, 2016, pp. 207–212.

[9] M. G. Xiao and C. Liu, Semantic relation classification via hierarchical recurrent neural network with attention, in *Proceedings of COLING, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, 2016, pp. 1254–1263.

[10] L. L. Wang, Z. Cao, G. de Melo, and Z. Y. Liu, Relation classification via multi-level attention CNNs, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, 2016, pp. 1298–1307.

[11] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, WA, USA, 2013, pp. 1631–1642.

[12] R. Socher, D. Q. Chen, C. D. Manning, and A. Y. Ng, Reasoning with neural tensor networks for knowledge base completion, in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, Lake Tahoe, NV, USA, 2013, pp. 464–469.

[13] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó. Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals, in *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, Boulder, CO, USA, 2009, pp. 94–99.

[14] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, Semantic compositionality through recursive matrix-vector spaces, in *Proceedings of 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Jeju Island, Korea, 2012, pp. 1201–1211.

[15] M. Yu, M. R. Gormley, and M. Dredze, Factor-based compositional embedding models, in *NIPS Workshop on Learning Semantics*, Montreal, Canada, 2014, pp. 95–101.

[16] Y. Xu, R. Jia, L. L. Mou, G. Li, Y. C. Chen, Y. Y. Lu, and Z. Jin, Improved relation classification by deep recurrent neural networks with data augmentation, in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, 2016.

[17] D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv: 1409.0473, 2014.

[18] X. P. Qiu and X. J. Huang, Convolutional neural tensor network architecture for community-based question answering, in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, Buenos Aires, Argentina, 2015, pp. 1305–1311.

[19] W. Z. Pei, T. Ge, and B. B. Chang, Max-margin tensor neural network for Chinese word segmentation, in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, MD, USA, 2014, pp. 293–303.

[20] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] W. Zaremba, I. Sutskever, and O. Vinyals, Recurrent neural network regularization, arXiv preprint arXiv: 1409.2329, 2014.

[22] M. T. Luong, H. Pham, and C. D. Manning, Effective approaches to attention-based neural machine translation, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv: 1301.3781, 2013.

[24] B. Rink and S. Harabagiu, UTD: Classifying semantic relations by combining lexical and semantic resources, in *Proceedings of the 5th International Workshop on Semantic Evaluation*, Uppsala, Sweden, 2010, pp. 256–259.

[25] Y. T. Shen and X. J. Huang, Attention-based convolutional neural network for semantic relation extraction, in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, 2016, pp. 2526–2536.

Runyan Zhang is currently a master student in China University of Mining and Technology. He got the bachelor degree from China University of Mining and Technology in 2016. His research interests include natural language processing and deep neural networks.

Fanrong Meng is a professor in China University of Mining and Technology. She got the PhD degree from China University of Mining and Technology in 1989. Her research interests include intelligent information processing, database technology, and data mining.

Yong Zhou is a professor in China University of Mining and Technology. He got the PhD degree from China University of Mining and Technology in 2001. His research interests include wireless sensor network and evolutionary computation.

Bing Liu is currently an associate professor in China University of Mining and Technology. He got the PhD degree from China University of Mining and Technology in 2008. His research interests include data mining, machine learning, and pattern recognition.