# Customized Attention Mechanism for Relation Classification

**Shirong Shen**†
School of Computer Science and Engineering, University of Electronic Science and Technology of China
Chengdu, China
1170743490@qq.com

**Yang Wen**†
School of Computer Science and Engineering, University of Electronic Science and Technology of China
Chengdu, China
201621060713@std.uestc.edu.cn

**Lijuan Zhou**
Natural Language Processing Laboratory, College of Information and Engineering, Zhengzhou University
Zhengzhou, China
1967331775@qq.com

**Hongyin Zan**
Natural Language Processing Laboratory, College of Information and Engineering, Zhengzhou University
Zhengzhou, China
iehyzan@zzu.edu.cn

## Abstract

This paper proposes a novel attention mechanism, customized attention mechanism, for relation classification. Based on the task requirements, logical constraints of the attention mechanism have been designed to control the learning process of network, which enhances the controllability in learning. Specifically, a logical constraint is an operation that affects the attention weight. This operation controls the weights of the attention mechanism through a generating function and a limiting function. Appropriate logical constraints improve the feature extraction ability and the robustness of the network, which are important supplements to the traditional neural network. Experiments show that our model and logical constraints have achieved a considerable improvement on the SemEval-2010 relation classification task, which outperforms the most advanced methods.

## 1 Introduction

Relational classification is an important topic of natural language processing. It aims to extract the semantic relationship between two entities in a sentence. For instance, in the sentence "[brandy] is distilled from fermented fruit [juices]", the entities 'brandy' and 'juices' are of relation **Entity-Origin**. Relational classification can be applied to semantic information extraction, questions answering and knowledge-based completion.

Traditional relation classification methods emphasize well-designed features (Rink and Harabagiu, 2010), or extend features using the kernel (Zelenko et al., 2003; Bunescu and Mooney, 2005), Rink and Harabagiu (2010) have achieved the best results by using support vector machines at the early stages. However, such methods are highly dependent on hand-craft linguistic features, which result in significant limitations and numerous preprocessing effort.

Nowadays, deep neural networks have achieved great success in relational classification task. A number of methods have been proposed for relation classification which use two architectures including convolutional neural networks (CNN) and recurrent neural networks (RNN). Socher et al. (2012) used an RNN-based architecture to generate compositional vector representations of sentences. Zeng et al. (2014) proposed a CNN network integrating with position embeddings to make up for the shortcomings of CNN missing contextual information. However, the features extracted by simple RNN and CNN are still not sensitive to location information which may miss core semantics. Thus, the attention mechanism has been designed to solve this problem.

Zhou et al. (2016) applied attention after bi-directional recurrent neural networks to extract relations between two entities. A multi-level attention CNN was proposed by Wang et al. (2016). Although such methods have achieved some improvements, the previous attention is still not good enough. Firstly, the traditional attention mechanism attempts to find an overall optimal attention scheme for all samples. This strategy can achieve good results in most scene, but it inevitably leads to poor performance in some samples that differ from global optimum. Secondly, the traditional attention mechanism is uncontrollable and lacks of logical constraints. These defects have inspired us that if some restrictions are added to the attention mechanism, we can guide the network to extract information that is useful for the task in the designated area. Thus, to achieve this target, we proposed a customized attention mechanism which can set logical constraints for attention weights.

In this paper, we have three key contributions:

1. We propose a novel attention mechanism, customized attention mechanism that restricts by manual constraints as a supplementary method for traditional feature extraction. Users can formulate constraints according to their needs or prior knowledge to affect attention weights, and then let the network focus on the information that users want it to focus on. The new attention mechanism can enhance the controllability and interpretability of learning. More importantly, customized attention mechanism can be applied to various deep learning tasks.
2. We design several effective constraints to make customized attention work and help our model achieve better results.
3. We evaluate our method on the SemEval-2010 relation classification task and achieve a state-of-the-art $F_1$-score of 89.3%.

## 2 Related Work

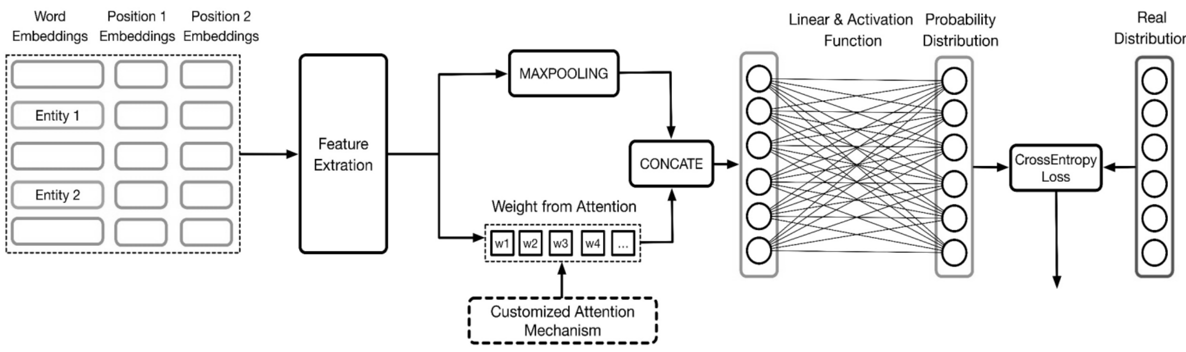
In recent years, deep neural networks have been found highly capable for solving NLP task like relational classification for their powerful feature extracting ability. An obvious idea is to use deep model like CNN to extract lexical and sentence level features from additional NLP resources (Zeng et al. 2014). Similar to traditional feature-based approaches, such methods could lead to limitation on performance due to confined linguistic knowledges. Thus, an alternative is to use sentence-level embedding to represent overall features of target text instead of handcrafted features set, together with syntactic dependency trees (Yu el al 2014) or Ranking CNN (Santos et al. 2015). Since then, although CNN have been widely used to capture local phrases connection, some observations have shown the RNN outperform CNN for its better ability to obtain long-term contextual information. Various RNN based architects have been proposed to address this advantage (Zhang et al., 2015; Xu et al., 2016; Miwa and Bansal, 2016). Actually, features extracted by both CNN and RNN are, in some way, compatible and unique. Thus, simply combine both CNNs and RNNs outputs have been proved improvement in relational classification task (Nguyen and Grishman 2015).

Different from average pooling or max pooling operations, which have been widely used for extracting sentence-level features from contextual outputs from CNNs or RNNs, attention mechanism produces a weighted output along time steps using a weight assignment activated by softmax function. Usage of attention mechanism helps models to evaluate different importance of outputs along time steps and have been proved profound improvement in several NLP task (Bahdanau et al., 2014; Rush et al., 2015; Yang et al., 2016). Because of its ideal performance, attention mechanism has been modified and improved for solving other tasks (Yang et al., 2017; Vaswani et al., 2017).

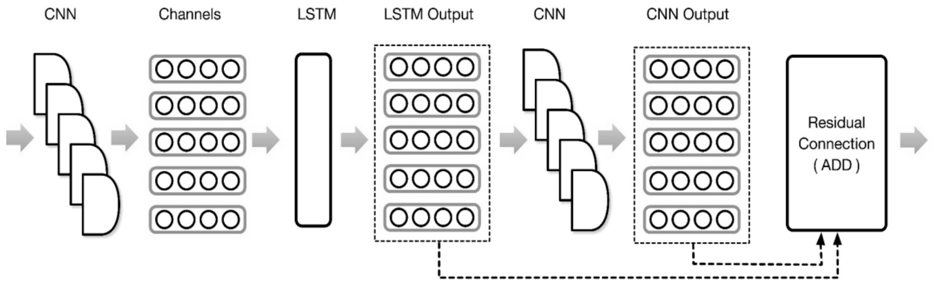
Although the previous works have achieved relatively satisfied results, we want the CNNs and RNNs could be architected into a single deep network instead of a parallel network to obtain both networks' advantages. And the attention strategy could be designed for certain linguistic priori. Our experiment section demonstrate that our model indeed achieves better performance in terms of the obtained F1 scores.

## 3 Model

The model of this paper consists of four parts: 1) embedding layer, a lookup table to map the input sequence to a matrix; 2) feature extraction module, a neural network to extract sentence-level features; 3) customized attention module, a special attention mechanism constructed as a supplementary solution for feature extraction pooling; 4) classification layer,



a



b

Figure 1: a. The overall architecture of our model; b. The structure of deep module

a full-connected layer to generate a probability distribution. Figure 1 depicts the overall architecture of our model. The details of the model will be introduced as follows:

### 3.1 Embedding layer

The input of the model is a word sequence $S$, $S = [w_1 \dots, w_{p_1}(e_1), \dots, w_{p_2}(e_1), \dots, w_l]$; $w_j$ is the j-th word in $S$, $e_1, e_2$ are two entities of $S$, $l$ is the length of $S$, $p_i$ $(i = 1,2)$ is the position of $e_i$. We represent each word in $S$ as a $d_w$-dimensional vector $E_{w_j}^{d_w}$ $(j = 1,2, \dots, l)$. In addition, we use the relative position between each word and the entities as extra features. The relative position $p_{i,j}$ between $w_j$ and $e_i$ is defined as $j - p_i$. For example, $S = [the, company(e_1), fabricates, plastic, chairs(e_2)]$, and the position relative to $e_2$ is $[-4, -3, -2, -1, 0]$. If the absolute value of the relative position is greater than the $T$, we replace it with $I$ (where $T$ is a positive integer, $I$ is an equivalent position marker to other integers). We represent each position as a $d_p$-dimensional vector $E_{p_{i,j}}^{d_p}$ $(i = 1,2; j = 1,2, \dots, l)$. Now we obtain the feature vector for each word: $E_j = [E_{w_j}^{d_w}; E_{p_{1,j}}^{d_p}; E_{p_{2,j}}^{d_p}]$, the sequence $S$ is represented as $E_S = [E_1', E_2', \dots, E_l']$.

### 3.2 Feature Extraction module

This part is designed to extract features from the input sequence that are favorable for relational classification. The input is $E_S$. In order to reflect the influence of the custom attention mechanism in different situations, we have designed two feature extraction modules: shallow module and depth module

#### 3.2.1 Shallow module

Shallow module is a single convolutional layer with window size 1 and 3. We pad the input to make the output of convolutional layer has the same feature length as input length $l$. Formally, the convolution operation in the article is defined as follows:

$$C_{i,j,k} = f_a\left(W_{coni,k} \cdot X_j + b_{coni,k}\right)$$
$$C_{i,j,k} = f_a\left(W_{coni,k} \cdot [X_{j-1}; X_j; X_{j+1}] + b_{coni,k}\right)$$

Where $X$ is a sequence input, $X_j$ is the element of the $j$-th position of $X$; $W_{coni,k}$ is the weight matrix of the $i$-th convolutional layer which window size is $k$, each convolutional layer we use two windows size $k_1, k_2$; $b_{coni,k}$ is the bias term of each hidden state vector, $f_a$ is a non-linear activation function ($tanh$ is used in our model). Then we concatenate them in one vector, $C_{i,j} = [C_{i,j,k_1}; C_{i,j,k_2}]$, a $2 * d_h$-

dimensional vector as the output of this layer. The input of this layer ($i = 0$) is $E$, $k_1 = 1, k_2 = 3$ and the feature that extracted from this module is:

$$F = [C_{0,1}, C_{0,2}, \dots, C_{0,l}]$$

### 3.2.2 Depth module

In addition to simple models, we want to design a complex and powerful model to verify the feasibility of attention mechanism. So, we use a three-tier network for feature extraction, which utilizes CNN, RNN and residual operation to improve feature extraction capabilities. The structure of the module is shown in Figure 1.

First, a convolutional layer is used to extract unigram information and local context information. As the definition of convolutional layer in last section, the input of this layer ($i = 1$) is $E$, $k_1 = 1, k_2 = 3$ and the output of this layer is:

$$C_1 = [C_{1,1}, C_{1,2}, \dots, C_{1,l}]$$

Then we use a Bi-LSTM (bidirectional Long Short-Term Memory) layer to extract sequence characteristic of language. The Bi-LSTM layer still produces a sequence with the length $l$, the vector of each position is the meaning of the word combination context of the current position which can be seen as the semantic feature of the word in this sentence (Normally, recurrent neural network is used to extract the overall information of the sequence, but in fact it is also able to extract the characteristics from the current context of each word. The final result of this operation depends on how to use its output).

The LSTM (Long Short-Term Memory) unit at $j$-th position consists of a collection of $d_h$-dimensional vectors: an input gate $i_j$, a forget gate $f_j$, an output gate $o_j$, a memory cell $c_j$, and a hidden state $h_j$. The unit receives an input vector $x_j$, the previous hidden state $h_{j-1}$ and the memory cell $c_{j-1}$ and calculates the new vectors using the following equations:

$$i_j = \sigma(W^{(i)} \cdot x_j + U^{(i)} \cdot h_{j-1} + b^{(i)})$$
$$f_j = \sigma(W^{(f)} \cdot x_j + U^{(f)} \cdot h_{j-1} + b^{(f)})$$
$$o_j = \sigma(W^{(o)} \cdot x_j + U^{(o)} \cdot h_{j-1} + b^{(o)})$$
$$u_j = f_a(W^{(u)} \cdot x_j + U^{(u)} \cdot h_{j-1} + b^{(u)})$$
$$c_j = i_j \odot u_j + f_j \odot c_{j-1}$$
$$h_j = o_j \odot f_a(c_j)$$

Where $\sigma$ denotes the logistic function, $\odot$ denotes element-wise multiplication, $W$ and $U$ are weight matrices, and $b$ are bias vectors. In the calculation process $x_j = C_{1,j}$. The hidden state vectors calculate the two directions' LSTM units corresponding to each position (denoted as $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$) as its output vectors, $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$ spliced into $R_j$,$R_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$ is a $2 * d_h$-dimensional vector as the output of Bi-LSTM in each position.

Next, the output of the Bi-LSTM is convolved to extract deeper information. Since each position is the meaning of the word in the current context, a more profound meaning can be extracted after another convolutional layer. The input of this layer ( $i = 1$ ) is $[R_1, R_2, \dots, R_l]$ , window size $k_1 = 1, k_2 = 3$. We get,

$$C_2 = [C_{2,1}, C_{2,2}, \dots, C_{2,l}]$$

Both the output of the second convolution layer and the output of the Bi-LSTM are used to make a difference to produce the final feature sequence $F$. The purpose of this operation is to extract relationship information other than word meaning.

$$F_j = C_{2,j} - R_j$$
$$F = [F_1, F_2, \dots, F_l]$$

### 3.2.3 Dimensionality reduction

To reduce feature dimensions, we use max pooling of $F$ as regular feature for classification. $F$ is a $l * (2d_h)$-dimensional matrix, the same position of each vector represents the same characteristics. The operation can be formulated as follows:

$$F_{max,i} = \max\{F_{1,i}, F_{2,i}, \dots, F_{l,i}\}$$
$$F_{max} = [F_{max,1}, F_{max,2}, \dots, F_{max,2*d_h}]$$

where $F_{max,i}$ is the $i$-th dimension of the output, $F_{j,i}$ ($j = 1,2, \dots, l$) is the $i$-th dimension of $F_j$.

## 3.3 Customized attention module

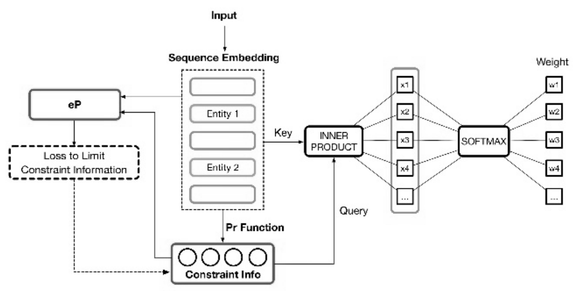


Figure 2: architecture of customized attention mechanism.

Customized attention is crafted as a supplement for feature extraction, a manual constraint is used to guide the network to focus on special information and provide logical constraint. The architecture is given in Figure 2.

Traditional feature extraction generates universal feature. Differently, the customized attention mechanism can guide the network to focus on the information as desired. We define a generating function $Pr$ to extract customized information of the statement and a limiting function $eP$ to limit $Pr$ to extract the information as we want. The structure consisting $eP$ and $Pr$ provide logical constraint for learning.

$$pr = Pr\ (E_S)$$
$$ep = eP(E_S, pr)$$

Then we use $pr$ and $E_S$ to calculate the weight of each position:

$$\begin{aligned} weight &= softmax(pr' \cdot [E_1, E_2, \dots, E_l]) \\ &= softmax(pr' \cdot E_S) \end{aligned}$$

Where $pr'$ is the transpose of $pr$. After that, we obtain the features that we want the network to focus on by weighting the $F$ by $weight$.

$$F_{att} = F \cdot weight'$$

$F_{att}$ will participate in the final classification task as an additional feature. According to the analysis in the previous section, each position of $F$ represents the deep semantic features of the current position. The position of $F$ is in one-to-one correspondence with the position of $E_S$, so $weight$ can increase the role of the contents we want to focus and make the feature extraction module be more sensitive to these contents.

This attention mechanism can be used in various deep learning tasks. Different manual constraints can be designed according to different logical constraints to design different $eP$ and $Pr$. This method does not require the extraction of new features, only needs to design logical constraints on real data at the input level and abstract it as a function. It can weaken the uncontrollability of the learning process and preserve the advantages of the original attention mechanism, in other words, $weight$ is determined by both logical constraints and task optimization.

In this paper, we use several different constraints to control our attention mechanism, the $Pr$ and $eP$ function of each constraints will be given in next section.

## 3.4 Classification layer

We concatenate $F_{att}$ and $F_{max}$ as input of the classification layer, $F_{final} = [F_{att}; F_{max}]$. Let the number of candidate relations be $K$, the output of this layer is a $K$-dimensional vector, each dimension represents the likelihood of the corresponding relationship.

We use a full-connected layer and a Softmax activation layer to generate final probability distributions:

$$out = softmax(W_f \cdot F_{final} + b_f)$$

$W_f$ is the weight and $b_f$ is the bias term of the full connection, and the $softmax$ is defined as:

$$y = softmax([x_1, x_2, \dots, x_n])$$
$$y_j = \frac{e^{x_j}}{\sum_{i=1}^{n} e^{x_i}}$$

Where $[x_1, x_2, \dots, x_n]$ and $y$ are the input and output of $softmax$.

## 3.5 Training process

We use the cross-entropy loss function to train our model and obtain the optimized result. Let $out$ be the result of input $S$ through the network

$$eC = \sum_{j=1}^{K} real_j \cdot ln(out_j)$$

$$real_j = \begin{cases} 1, & S\ belongs\ to\ the\ jth\ relationship \\ 0, & others \end{cases}$$

The final loss function is defined as,

$$e = eC + \lambda \cdot ep$$

where $\lambda$ is a hyper-parameter used to control the influence of manual constraints on the attention mechanism.

For a single data sample, the training objective is $e$ and we use Adam (Adaptive Moment Estimation) to train network parameters.

## 4 Experiments

We evaluate our model on the SemEval-2010 Task 8 dataset. The dataset contains 8000 sentences for training, and 2717 for testing. The dataset has $K = 10$ candidate relations, as follow:

- Cause-Effect
- Component-Whole
- Content-Container
- Entity-Destination
- Entity-Origin
- Message-Topic
- Member-Collection
- Instrument-Agency
- Product-Agency
- Other

All baseline and our models use the official macro-averaged $F_1$-score to evaluate model performance.

Our customized attention mechanism has been tested under several different constraints with shallow module and depth module.

In our experiment, word embedding dimension was $d_w = 50$ as used in (Turian et al. 2010); position embedding dimension was $d_p = 25$; relative position boundary was $T = 60$; dropout of the embedding was 0.3; hidden size of each layer was $d_h = 100$; Parameter $\lambda = 0.1$; Adam has been applied for optimization with initial learning rate $0.001$.

## 4.1 Constraints of customized attention

The purpose of the constraints is to guide the network to pay more attention to certain words or positions. In the current task, the input consists of two entities and their context. Each word, including entity, may have important implication for the mission. In this paper, the words in the input sentence are divided into three categories: *entities*, *core words* and *marginal words*. *Entities* refer to the two entities given in the task and *core words* are words which are located around the designated entity pair (excluding *entities*), *marginal words* are the words other than *entities* and *core words*. Except that the *entities* are shown given, the other two categories are defined by generating. In our experiment, through the combinations of the above three types of inputs, customized attention guides the network to pay attention to four different areas. All $Pr$ functions have been defined as a linear transformation to entities with a non-linear activation function $f_a$(we use $tanh$ this paper). Formally,

$$\begin{aligned} pr &= Pr(E_S) \\ &= f_a(W_{pr} \cdot \left[E_{w_{p1}}^{d_w}; E_{w_{p2}}^{d_w}\right] + b_{pr}) \end{aligned}$$

where $W_{pr}$ is the weight for the linear transformation, $b_{pr}$ is the bias term for the hidden vector.

We use four different $eP$ functions to limit the vector calculated by $Pr$. In this paper, $eP$ reaches this goal by generating a target vector and controlling(minimizing) the cosine similarity between this vector and $pr$. The cosine similarity is defined as follow:

$$\cos(a, b) = \frac{a \cdot b}{\|a\|\|b\| + \varepsilon}$$

$a, b$ are two vectors with the same dimensions, $\varepsilon$ is a positive number prevents the denominator is 0, $\|\cdot\|$ is 2-norm of vector. We use the direction of the vector to measure the word category, not related to the norm of the vector. In this way, it is possible to distinguish between different categories of words without having a great influence on the original word vectors.

We set several $eP$ as follow:

**a) Entities & Marginal words:**

$$\begin{aligned} ep &= eP(E_S, pr) \\ &= \cos\left(pr, \frac{\sum_{i \neq p_1, p_2}^{l} E_{w_i}^{d_w}}{l-2}\right) \end{aligned}$$

Target vector is $\frac{\sum_{i \neq p_1, p_2}^{l} E_{w_i}^{d_w}}{l-2}$, the vector is the mean of the word vectors of $S$ except $e_1$, $e_2$. We use this vector approximate the *core words*. From a statistical point of view, this vector has a smaller angle with *core words* (high frequency words) than *marginal words* and *entities*. When we decrease $ep$ during training, $pr$ will stay away from *core words*. So, this constraint makes our model pay more attention to *entities* and *marginal words* in $S$.

**b) Marginal words:**

$$\begin{aligned} ep &= eP(E_S, pr) \\ &= \cos\left(pr, \frac{\sum_{i}^{l} E_{w_i}^{d_w}}{l}\right) \end{aligned}$$

Target vector is $\frac{\sum_{i}^{l} E_{w_i}^{d_w}}{l}$ is the mean of the all word vectors of $S$. Since the *entities* must appear in the statement, the *entities* can be seen as a high frequency words, the target vector has a smaller angle with *core words* and *entities* than *marginal words*. It can represent the core words and entities, $pr$ will stay away from them when we decrease $ep$ during training.

**c) Core words:**

$$\begin{aligned} ep &= eP(E_S, pr) \\ &= \cos\left(pr, -\frac{\sum_{i \neq p_1, p_2}^{l} E_{w_i}^{d_w}}{l-2}\right) \end{aligned}$$

Target vector is $-\frac{\sum_{i \neq p_1, p_2}^{l} E_{w_i}^{d_w}}{l-2}$, when we reduce $ep$, we get the opposite effect as **a)**.

**d) Entities & Core words:**

$$\begin{aligned} ep &= eP(E_S, pr) \\ &= \cos\left(pr, -\frac{\sum_{i}^{l} E_{w_i}^{d_w}}{l}\right) \end{aligned}$$

This operation will make $pr$ close to the core words, and entities.

## 4.2 Result and analysis

Table 1 compares our model (uses depth module) with previous state-of-the-art methods. We observe that our model achieves new state-of-the-art result on this dataset. Customized attention strengthened the extraction of relations between *marginal words* and *entities*, helping the network go further.

| Classifier | $F_1$ |
|---|---|
| SVM (Rink and Harabagiu, 2010) | 82.2 |
| MVRNN (Socher et al., 2012) | 82.4 |
| FCM (Yu et al., 2014) | 83.0 |
| depLCNN + NS (Xu et al., 2015a) | 85.6 |
| DRNNs (Xu et al., 2016) | 85.8 |
| BRCNN (Cai et al., 2016) | 86.3 |
| Att-Pooling-CNN (Wang et al., 2016) | 88.0 |
| *Our model* | |
| (Depth module) without customized attention | 88.9 |
| (Depth module) Att: Entity & Marginal words | **89.3** |

Table 1：comparison with other models

To demonstrate the effect of different constraints on customized attention model, we did experiments using different rules. Table 2 and Table 3 illustrate the experimental results. In particular, experiments in Table 2 use depth module for feature extraction and in Table 3 use shallow module to extract feature. In Table 3 we also compared our attention mechanism with Multi-Level Attention (Wang et al., 2016), only differ from our model in the attention mechanism. When the feature extraction module is different, the influence of different customized attention mechanisms is diverse. The most important message in the experiments is a suitable manual constraint will certainly help the network extract more effective, robust features and achieves a better score than traditional attention mechanism does.

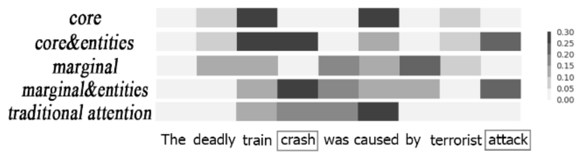


Figure 3: attention weights of certain sentence with various constraints.

We chose a sentence *'The deadly train [crash] was caused by terrorist [attack]'* and visualized weights of the attention under different constraints in Figure 3, the entities is *crash* and *attack*. When we focus on *entities* and *core words*, our attentions have given entities and high frequency words heavy weight. In this sentence, *core words* are *train* and *caused*, representing important nouns and verbs. As shown in Table 3, the performance dropped after we applied these two constraints, which indicated that the network had already paid attention to these words without using the attention mechanism, and effectively used them for classification. Thus, such attentions are redundant. Follow this conclusion, when we focus on *entities* and *marginal words* which are easily overlooked, we can use more abundant information to classify. In the current sentence, *marginal words* are *was* and *by*, representing connection information between words. The experimental results also support our inferences. When we focus on *entities* and *marginal words*, scores are improved (88.9% to 89.3% with depth module, 87.7% to 88.3% with shallow module). And as we increase the weight of these positions, the network will extract better features for these positions (since in the back-propagation process, the influence of these positions on the overall network parameters are increased), the feature extraction module learns more during training.

| Focus on information | $F_1$ |
|---|---|
| Without customized attention | 88.9 |
| Att: Entity & Core words | 88.6 |
| Att: Core words | 88.9 |
| Att: Marginal words | 89.0 |
| Att: Entity & Marginal words | **89.3** |

Table 2: Comparison of different customized attention mechanisms with depth module.

| Focus on information | $F_1$ |
|---|---|
| Without customized attention | 87.7 |
| Att-Pooling-CNN (Wang et al., 2016) | 88.0 |
| Att: Entity & Core words | 87.9 |
| Att: Core words | 88.4 |
| Att: Marginal words | 88.2 |
| Att: Entity & Marginal words | **88.5** |

Table 3: Comparison of different customized attention mechanisms with shallow module.

## 5 Conclusion

In this paper, we propose a customized attention mechanism to improve the performance of relation classification. We use two feature extraction modules: a shallow one uses a single CNN, a depth module uses two CNN layers and an RNN layer, can effectively extract classification features. The performance of the deep feature extraction module with customized attention mechanism is better than the state-of-the-art methods. Most importantly, the custom attention mechanism achieved better performance in every situation. Experiments show that using both shallow module and depth module, appropriate manual constraints will greatly enhance the performance of the model. The formulation of manual constraints entirely depends on your own needs. You can choose the area you want to focus on and guide the network to learn more about it. The customized attention mechanism can make up for the lack of direct learning, improve the efficiency of feature extraction, and enhance the robustness of the model. It is worth promoting.

## Reference

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language*, pages 724–731.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, Bo Xu. 2016. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 207–212

Linlin Wang, Zhu Cao, Gerard de Melo and Zhiyuan Liu. 2016. Relation Classification via Multi-Level Attention CNNs. In *proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1298–1307.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natura.*

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation pages*, pages 73–78.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint arXiv:1601.03651.*

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. arXiv preprint arXiv:1601.00770.

Thien Huu Nguyen and Ralph Grishman. 2015. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926.*

Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. Computer Science.

Rush Alexander M, Chopra Sumit and Weston, Jason. 2015. A neural attention model for abstractive sentence summarization. Computer Science.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng and Alex Smola. 2016. Stacked Attention Networks for Image Question Answering. IEEE Conference on Computer Vision and Pattern Recognition, pages 21-29. IEEE Computer Society.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2017. Hierarchical Attention Networks for Document Classification. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1480-1489.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin. 2017. Attention is all you need. arXiv preprint arXiv: 1706.03762.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. *Proceedings of EMNLP 2015*.

Rui Cai, Xiaodong Zhang and Houfeng Wang. 2016. Bidirectional Recurrent Convolutional Neural Network for Relation Classification. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 756–765.