Bidirectional Recurrent Convolutional Neural Network for Relation Classification

Rui Cai, Xiaodong Zhang and **Houfeng Wang**[*]
Key Laboratory of Computational Linguistics (Ministry of Education),
School of EECS, Peking University, Beijing, 100871, China
Collaborative Innovation Center for Lanuage Ability, Xuzhou, Jiangsu, 221009, China
{cairui, zxdcs, wanghf}@pku.edu.cn

Abstract

Relation classification is an important semantic processing task in the field of natural language processing (NLP). In this paper, we present a novel model BRCNN to classify the relation of two entities in a sentence. Some state-of-the-art systems concentrate on modeling the shortest dependency path (SDP) between two entities leveraging convolutional or recurrent neural networks. We further explore how to make full use of the dependency relations information in the SDP, by combining convolutional neural networks and two-channel recurrent neural networks with long short term memory (LSTM) units. We propose a bidirectional architecture to learn relation representations with directional information along the SDP forwards and backwards at the same time, which benefits classifying the direction of relations. Experimental results show that our method outperforms the state-of-the-art approaches on the SemEval-2010 Task 8 dataset.

1 Introduction

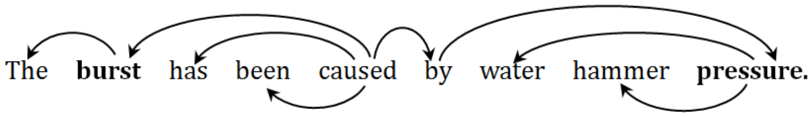
Relation classification aims to classify the semantic relations between two entities in a sentence. For instance, in the sentence "The [burst]$_{e_1}$ has been caused by water hammer [pressure]$_{e_2}$", entities *burst* and *pressure* are of relation Cause-Effect(e_2, e_1). Relation classification plays a key role in robust knowledge extraction, and has become a hot research topic in recent years.

Nowadays, deep learning techniques have made significant improvement in relation classification, compared with traditional relation classification approaches focusing on designing effective features (Rink and Harabagiu, 2010) or kernels (Zelenko et al., 2003; Bunescu and Mooney, 2005) Although traditional approaches are able to exploit the symbolic structures in sentences, they still suffer from the difficulty to generalize over the unseen words. Some recent works learn features automatically based on neural networks (NN), employing continuous representations of words (word embeddings). The NN research for relation classification has centered around two main network architectures: convolutional neural networks and recursive/recurrent neural networks. Convolutional neural network aims to generalize the local and consecutive context of the relation mentions, while recurrent neural networks adaptively accumulate the context information in the whole sentence via memory units, thereby encoding the global and possibly unconsecutive patterns for relation classification. Socher et al. (2012) learned compositional vector representations of sentences with a recursive neural network. Kazuma et al. (2013) proposed a simple customizaition of recursive neural networks. Zeng et al. (2014) proposed a convolutional neural network with position embeddings.

Recently, more attentions have been paid to modeling the shortest dependency path (SDP) of sentences. Liu et al. (2015) developed a dependency-based neural network, in which a convolutional neural network has been used to capture features on the shortest path and a recursive neural network is designed to model subtrees. Xu et al. (2015b) applied long short term memory (LSTM) based recurrent neural networks (RNNs) along the shortest dependency path. However, SDP is a special structure in which every two neighbor words are separated by a dependency relations. Previous works treated dependency relations in the same

[*] Corresponding author

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 756–765,
Berlin, Germany, August 7-12, 2016. ©2016 Association for Computational Linguistics



Realtion	Shortest Dependency Path
Cause-Effect(e_2,e_1)	burst ←nsubjpass— caused —prep→ by —pobj→ pressure
Cause-Effect(e_1,e_2)	pressure ←pobj— by ←prep— caused —nsubjpass→ burst

Figure 1: The shortest dependency path representation for an example sentence from SemEval-08.

way as words or some syntactic features like part-of-speech (POS) tags, because of the limitations of convolutional neural networks and recurrent neural networks. Our first contribution is that we propose a recurrent convolutional neural network (RCNN) to encode the global pattern in SDP utilizing a two-channel LSTM based recurrent neural network and capture local features of every two neighbor words linked by a dependency relation utilizing a convolution layer.

We further observe that the relationship between two entities are directed. For instance, Figure 1 shows that the shortest path of the sentence "The [burst]$_{e_1}$ has been caused by water hammer [pressure]$_{e_2}$." corresponds to relation Cause-Effect(e_2, e_1). The SDP of the sentence also corresponds to relation Cause-Effect(e_2, e_1), where e_1 refers to the entity at front end of SDP and e_2 refers to the entity at back end of SDP, and the inverse SDP corresponds to relation Cause-Effect(e_1, e_2). Previous work (Xu et al., 2015b) simply transforms a (K+1)-relation task into a $(2K + 1)$ classification task, where 1 is the *Other* relation and K is the number of directed relations. Besides, the recurrent neural network is a biased model, where later inputs are more dominant than earlier inputs. It could reduce the effectiveness when it is used to capture the semantics of a whole shortest dependency path, because key components could appear anywhere in a SDP rather than the end.

Our second contribution is that we propose a bidirectional recurrent convolutional neural networks (BRCNN) to learn representations with bidirectional information along the SDP forwards and backwards at the same time, which also strengthen the ability to classifying directions of relationships between entities. Experimental results show that the bidirectional mechanism significantly improves the performance.

We evaluate our method on the SemEval-2010 relation classification task, and achieve a state-of-the-art F_1-score of 86.3%.

2 The Proposed Method

In this section, we describe our method in detail. Subsection 2.1 provides an overall picture of our BCRNN model. Subsection 2.2 presents the rationale of using SDPs and some characteristics of SDP. Subsection 2.3 describes the two-channel recurrent neural network, and bidirectional recurrent convolutional neural network is introduced in Subsection 2.4. Finally, we present our training objective in Subsection 2.5.

2.1 Framework

Our BCRNN model is used to learn representations with bidirectional information along the SDP forwards and backwards at the same time. Figure 2 depicts the overall architecture of the BRCNN model.

Given a sentence and its dependency tree, we build our neural network on its SDP extracted from the tree. Along the SDP, two recurrent neural networks with long short term memory units are applied to learn hidden representations of words and dependency relations respectively. A convolution layer is applied to capture local features from hidden representations of every two neighbor words and the dependency relations between them. A max pooling layer thereafter gathers information from local features of the SDP or the inverse SDP. We have a $softmax$ output layer after pooling layer for classification in the unidirectional model RCNN.

On the basis of RCNN model, we build a bidirectional architecture BRCNN taking the SDP and the inverse SDP of a sentence as input. During the training stage of a (K+1)-relation task, two fine-grained $softmax$ classifiers of RCNNs do a $(2K + 1)$-class classification respectively. The pooling layers of two RCNNs are concatenated and a coarse-grained $softmax$ output layer is followed to do a $(K + 1)$-class classification. The final (2K+1)-class distribution is the combination of two (2K+1)-class distributions provided by fine-grained classifiers respectively during the testing stage.

2.2 The Shortest Dependency Path

If e_1 and e_2 are two entities mentioned in the same sentence such that they are observed to be in a relationship R, the shortest path between e_1 and e_2 condenses most illuminating information for the relationship $R(e_1, e_2)$. It is because (1) if entities e_1 and e_2 are arguments of the same predicate, the shortest path between them will pass through the predicate; (2) if e_1 and e_2 belong to different predicate-argument structures that share a common argument, the shortest path will pass through this argument.

Bunescu and Mooney (2005) first used shortest dependency paths between two entities to capture the predicate-argument sequences, which provided strong evidence for relation classification. Xu et al. (2015b) captured information from the sub-paths separated by the common ancestor node of two entities in the shortest paths. However, the shortest dependency path between two entities is usually short (~4 on average) , and the common ancestor of some SDPs is e_1 or e_2, which leads to imbalance of two sub-paths.

We observe that, in the shortest dependency path, each two neighbor words w_a and w_b are linked by a dependency relation r_{ab}. The dependency relations between a governing word and its children make a difference in meaning. Besides, if we inverse the shortest dependency path, it corresponds to the same relationship with an opposite direction. For example , in Figure 1, the shortest path is composed of some sub-structure like "$burst \xrightarrow{nsubjpass} caused$". Following the above intuition, we design a bidirectional recurrent convolutional neural network, which can capture features from the local substructures and inversely at the same time.

2.3 Two-Channel Recurrent Neural Network with Long Short Term Memory Units

The recurrent neural network is suitable for modeling sequential data, as it keeps hidden state vector h, which changes with input data at each step accordingly. We make use of words and dependency relations along the SDP for relations classification (Figure 2). We call them *channels* as these information sources do not interact during recurrent propagation. Each word and dependency relation in a given sentence is mapped to a real-valued vector by looking up in a embedding table. The embeddings of words are trained on a large corpus unsupervisedly and are thought to be able to capture their syntactic and semantic information, and the embeddings of dependency relations are initialized randomly.

The hidden state h_t, for the t-th input is a function of its previous state h_{t-1} and the embedding x_t of current input. Traditional recurrent networks have a basic interaction, that is, the input is linearly transformed by a weight matrix and nonlinearly squashed by an activation function. Formally, we have

$$h_t = f(W_{in} \cdot x_t + W_{rec} \cdot h_{t-1} + b_h) \qquad (1)$$

where W_{in} and W_{rec} are weight matrices for the input and recurrent connections, respectively. b_h is a bias term for the hidden state vector, and f a non-linear activation function.

It was difficult to train RNNs to capture long-term dependencies because the gradients tend to either vanish or explode. Therefore, some more sophisticated activation function with gating units were designed. Long short term memory units are proposed in Hochreiter and Schmidhuber (1997) to overcome this problem. The main idea is to introduce an adaptive gating mechanism, which decides the degree to which LSTM units keep the previous state and memorize the extracted features of the current data input. Many LSTM variants have been proposed. We adopt in our method a variant introduced by Zaremba and Sutskever (2014). Concretely, the LSTM-based recurrent neural network comprises four components: an input gate i_t, a forget gate f_t, an output gate o_t, and a memory cell c_t.

First, we compute the values for i_t, the input gate, and g_t the candidate value for the states of

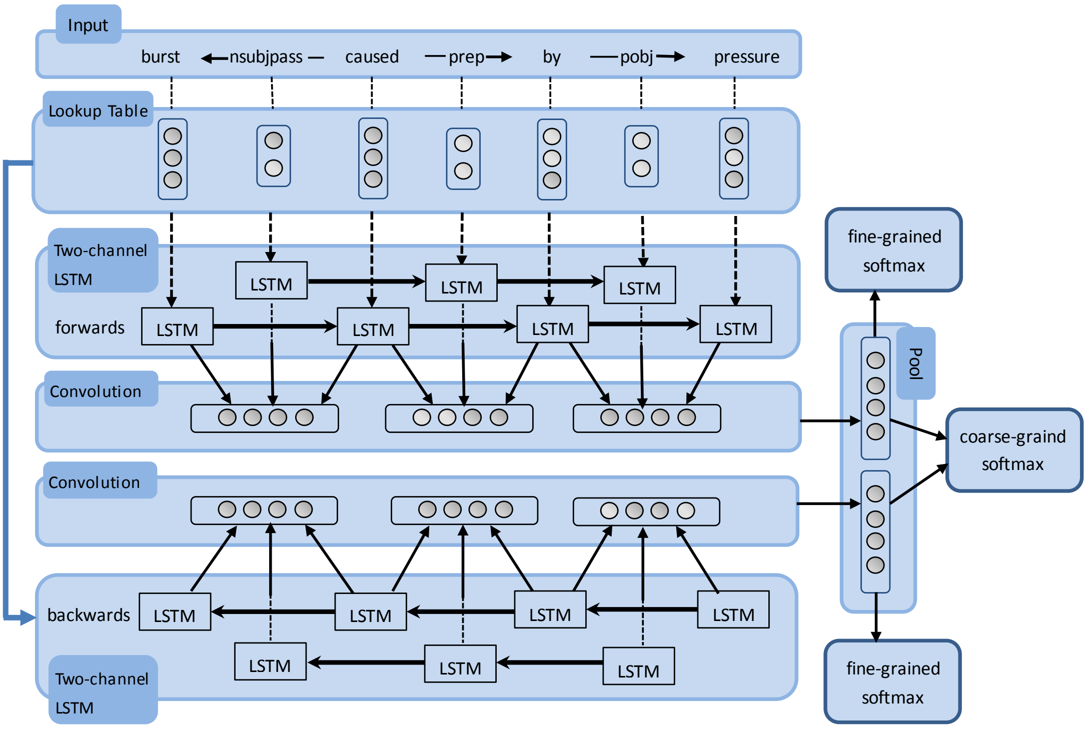


Figure 2: The overall architecture of BRCNN. Two-Channel recurrent neural networks with LSTM units pick up information along the shortest dependency path, and inversely at the same time. Convolution layers are applied to extract local features from the dependency units.

the memory cells at time t:

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \quad (2)$$

$$g_t = tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c) \quad (3)$$

Second, we compute the value for f_t, the activations of the memory cells' forget gates at time t:

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \quad (4)$$

Given the value of the input gate activations i_t, the forget gate activation f_t and the candidate state value g_t, we can compute c_t the memory cells' new state at time t:

$$c_t = i_t \otimes g_t + f_t \otimes c_{t-1} \quad (5)$$

With the new state of the memory cells, we can compute the value of their output gates and, subsequently, their outputs:

$$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \quad (6)$$

$$h_t = o_t \otimes tanh(c_t) \quad (7)$$

In the above equations, σ denotes a sigmoid function; $\otimes$ denotes element-wise multiplication.

2.4 Bidirectional Recurrent Convolutional Neural Network

We observe that a governing word w_a and its children w_b are linked by a dependency relation r_{ab}, which makes a difference in meaning. For example, "*kills* $\xrightarrow{nsubj}$ *it*" is distinct from "*kills* $\xrightarrow{dobj}$ *it*". The shortest dependency path is composed of many substructures like "$w_a \xrightarrow{r_{ab}} w_b$", which are hereinafter referred to as "*dependency unit*". Hidden states of words and dependency relations in the SDP are obtained, utilizing two-channel recurrent neural network. The hidden states of w_a, w_b and r_{ab} are h_a, h_b and h'_{ab}, and the hidden state of the dependency unit d_{ab} is $[h_a \oplus h'_{ab} \oplus h_b]$, where $\oplus$ denotes concatenate operation. Local features L_{ab} for the dependency unit d_{ab} can be extracted, utilizing a convolution layer upon the two-channel recurrent neural network . Formally, we have

$$L_{ab} = f(W_{con} \cdot [h_a \oplus h'_{ab} \oplus h_b] + b_{con}) \quad (8)$$

where W_{con} is the weight matrix for the convolution layer and b_{con} is a bias term for the hidden state vector. f is a non-linear activation function(*tanh* is used in our model). A pooling layer thereafter gather global information G from local features of dependency units, which is defined as

$$G = \max_{d=1}^{D} L_d \tag{9}$$

where the max function is an element-wise function, and D is the number of dependency units in the SDP.

The advantage of two-channel recurrent neural network is the ability to better capture the contextual information, adaptively accumulating the context information the whole path via memory units. However, the recurrent neural network is a biased model, where later inputs are more dominant than earlier inputs. It could reduce the effectiveness when it is used to capture features for relation classification, for the entities are located at both ends of SDP and key components could appear anywhere in a SDP rather than at the end. We tackle the problem with Bidirectional Convolutional Recurrent Neural Network.

On the basis of observation, we make a hypothesis that SDP is a symmetrical structure. For example, if there is a forward shortest path $\overrightarrow{S}$ which corresponds to relation $R_x(e_1, e_2)$, the backward shortest path $\overleftarrow{S}$ can be obtained by inversing $\overrightarrow{S}$, and $\overleftarrow{S}$ corresponds to $R_x(e_2, e_1)$, and both $\overrightarrow{S}$ and $\overleftarrow{S}$ correspond to relation R_x.

As shown in Figure 2, two RCNNs pick up information along $\overrightarrow{S}$ and $\overleftarrow{S}$, obtaining global representations $\overrightarrow{G}$ and $\overleftarrow{G}$. A representation with bidirectional information is obtained by concatenating $\overrightarrow{G}$ and $\overleftarrow{G}$. A coarse-grained $softmax$ classifier is used to predict a (K+1)-class distribution y. Formally,

$$y = softmax(W_c \cdot [\overleftarrow{G} \oplus \overrightarrow{G}] + b_c) \tag{10}$$

Where W_c is the transformation matrix and b_c is the bias vector. Coarse-grained classifier makes use of representation with bidirectional information ignoring the direction of relations, which learns the inherent correlation between the same directed relations with opposite directions, such as $R_x(e_1, e_2)$ and $R_x(e_2, e_1)$.

Two fine-grained $softmax$ classifiers are applied to $\overrightarrow{G}$ and $\overleftarrow{G}$ with linear transformation to give the (2K+1)-class distribution $\overrightarrow{y}$ and $\overleftarrow{y}$ respectively. Formally,

$$\overrightarrow{y} = softmax(W_f \cdot \overrightarrow{G} + b_f) \tag{11}$$

$$\overleftarrow{y} = softmax(W_f \cdot \overleftarrow{G} + b_f) \tag{12}$$

where W_f is the transformation matrix and b_f is the bias vector. Classifying $\overrightarrow{S}$ and $\overleftarrow{S}$ respecitvely at the same time can strengthen the model ability to judge the direction of relations.

2.5 Training Objective

The $(K + 1)$-class $softmax$ classifier is used to estimate probability that $\overrightarrow{S}$ and $\overleftarrow{S}$ are of relation R . The two $(2K + 1)$-class $softmax$ classifiers are used to estimate the probability that $\overrightarrow{S}$ and $\overleftarrow{S}$ are of relation $\overrightarrow{R}$ and $\overleftarrow{R}$ respectively. For a single data sample, the training objective is the penalized cross-entropy of three classifiers, given by

$$\begin{aligned} J = & \sum_{i=1}^{2K+1} \overrightarrow{t}_i \log \overrightarrow{y}_i + \sum_{i=1}^{2K+1} \overleftarrow{t}_i \log \overleftarrow{y}_i \\ & + \sum_{i=1}^{K} t_i \log y_i + \lambda \cdot \|\theta\|^2 \end{aligned} \tag{13}$$

where $t \in \mathbb{R}^{K+1}$, $\overrightarrow{t}$ and $\overleftarrow{t} \in \mathbb{R}^{2K+1}$, indicating the one-hot represented ground truth. y, $\overrightarrow{y}$ and $\overleftarrow{y}$ are the estimated probabilities for each class described in section 2.4. θ is the set of model parameters to be learned, and λ is a regularization coefficient.

For decoding (predicting the relation of an unseen sample), the bidirectional model provides the (2K+1)-class distribution $\overrightarrow{y}$ and $\overleftarrow{y}$. The final (2K+1)-class distribution y_{test} becomes the combination of $\overrightarrow{y}$ and $\overleftarrow{y}$. Formally,

$$y_{test} = \alpha \cdot \overrightarrow{y} + (1 - \alpha) \cdot z(\overleftarrow{y}) \tag{14}$$

where α is the fraction of the composition of distributions, which is set to the value 0.65 according to the performance on validation dataset. During the implementation of BRCNN, elements in two class distributions at the same position are not corresponding, e.g. Cause-Effect(e_1, e_2) in $\overrightarrow{y}$ should correspond to Cause-Effect(e_2, e_1) in $\overleftarrow{y}$. We apply a function z to transform $\overleftarrow{y}$ to a corresponding forward distribution like $\overrightarrow{y}$.

3 Experiments

3.1 Dataset

We evaluated our BRCNN model on the SemEval-2010 Task 8 dataset, which is an established benchmark for relation classification (Hendrickx et al., 2010). The dataset contains 8000 sentences for training, and 2717 for testing. We split 800 samples out of the training set for validation.

Classifier	Additional Information	F_1
SVM (Rink and Harabagiu, 2010)	POS, WordNet, Prefixes and other morphological features, dependency parse, Levin classed, PropBank, FanmeNet, NomLex-Plus, Google *n*-gram, paraphrases, TextRunner	82.2
RNN	Word embeddings	74.8
(Socher et al., 2011)	+ POS, NER, WordNet	77.6
MVRNN	Word embeddings	79.1
(Socher et al., 2012)	+ POS, NER, WordNet	82.4
CNN	Word embeddings	69.7
(Zeng et al., 2014)	+ word position embeddings, WordNet	82.7
FCM	Word embeddings	80.6
(Yu et al., 2014)	+ dependency parsing, NER	83.0
CR-CNN	Word embeddings	82.8
(dos Santos et al., 2015)	+ word position embeddings	84.1
SDP-LSTM	Word embeddings	82.4
(Xu et al., 2015b)	+ POS + GR + WordNet embeddings	83.7
DepNN	Word embeddings, WordNet	83.0
(Liu et al., 2015)	Word embeddings, NER	83.6
depLCNN	Word embeddings, WordNet, word around nominals	83.7
(Xu et al., 2015a)	+ negative sampling from NYT dataset	85.6
BRCNN	Word embeddings	85.4
(Our Model)	+ POS, NER, WordNet embeddings	**86.3**

Table 1: Comparison of relation classification systems.

The dataset has (K+1)=10 distinguished relations, as follows.

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

The former K=9 relations are directed, whereas the Other class is undirected, we have (2K+1)=19 different classes for 10 relations. All baseline systems and our model use the official macro-averaged F_1-score to evaluate model performance. This official measurement excludes the Other relation.

3.2 Hyperparameter Settings

In our experiment, word embeddings were 200-dimensional as used in (Yu et al., 2014), trained on Gigaword with word2vec (Mikolov et al., 2013). Embeddings of relation are 50-dimensional and initialized randomly. The hidden layers in each channel had the same number of units as their embeddings (200 or 50). The convolution layer was 200-dimensional. The above values were chosen according to the performance on the validation dataset.

As we can see in Figure 1, dependency relation r "$\xrightarrow{prep}$" in $\overrightarrow{S}$ becomes r^{-1} "$\xleftarrow{prep}$" in $\overleftarrow{S}$. Experiment results show that, the performance of BRCNN is improved if r and r^{-1} correspond to different relations embeddings rather than a same embedding. We notice that dependency relations contain much fewer symbols than the words contained in the vocabulary, and we initialize the embeddings of dependency relations randomly for they can be adequately tuned during supervised training.

We add l_2 penalty for weights with coefficient 10^{-5}, and dropout of embeddings with rate 0.5. We applied AdaDelta for optimization (Zeiler, 2012), where gradients are computed with an adaptive learning rate.

3.3 Results

Table 1 compares our BRCNN model with other state-of-the-art methods. The first entry in the table presents the highest performance achieved by traditional feature-based methods. Rink and Harabagiu. (2010) fed a variety of handcrafted features to the SVM classifier and achieve an F_1-score of 82.2%.

Recent performance improvements on this dataset are mostly achieved with the help of neural networks. Socher et al. (2012) built a recursive neural network on the constituency tree and achieved a comparable performance with Rink and Harabagiu. (2010). Further, they extended their recursive network with matrix-vector interaction and elevated the F_1 to 82.4%. Xu et al. (2015b) first introduced a type of gated recurrent neural network (LSTM) into this task and raised the F_1-score to 83.7%.

From the perspective of convolution, Zeng et al. (2014) constructed a CNN on the word sequence; they also integrated word position embeddings, which helped a lot on the CNN architecture. dos Santos et al. (2015) proposed a similar CNN model, named CR-CNN, by replacing the common *softmax* cost function with a ranking-based cost function. By diminishing the impact of the Other class, they have achieved an F_1-score of 84.1%. Along the line of CNNs, Xu et al. (2015a) designed a simple negative sampling method, which introduced additional samples from other corpora like the NYT dataset. Doing so greatly improved the performance to a high F_1-score of 85.6%. Liu et al. (2015) proposed a convolutional neural network with a recursive neural network designed to model the subtrees, and achieve an F_1-score of 83.6%.

Without the use of neural networks, Yu et al. (2014) proposed a Feature-based Compositional Embedding Model (FCM), which combined unlexicalized linguistic contexts and word embeddings. They achieved an F_1-score of 83.0%.

We make use of three types of information to improve the performance of BRCNN: POS tags, NER features and WordNet hypernyms. Our proposed BRCNN model yields an F_1-score of 86.3%, outperforming existing competing approaches. Without using any human-designed features, our model still achieve an F_1-score of 85.4%, while the best performance of state-of-the-art methods is 84.1% (dos Santos et al., 2015).

3.4 Analysis

Table 2 compares our RCNN model with CNNs and RNNs.

Model	F_1
CNN	81.8
LSTM	76.6
Two-channel LSTM	81.5
RCNN	**82.4**

Table 2: Comparing RCNN with CNNs and RNNS.

For a fair comparison, hyperparameters are set according to the performance on validation dataset as BRCNN . CNN with embeddings of words, positions and dependency relations as input achieves an F_1-score of 81.8%. LSTM with word embeddings as input only achieves an F_1-score of 76.6%, which proves that dependency relations in SDPs play an important role in relation classification. Two-channel LSTM concatenates the pooling layers of words and dependency relations along the shortest dependency path, achieves an F_1-score of 81.5% which is still lower than CNN. RCNN captures features from dependency units by combining the advantages of CNN and RNN, and achieves an F_1-score of 82.4%.

Model	**Input**	F_1
RCNN	$\overrightarrow{S}$ of all relations	82.4
Bi-RCNN	$\overrightarrow{S}$ and $\overleftarrow{S}$ of all relations	81.2
Bi-RCNN	$\overrightarrow{S}$ and $\overleftarrow{S}$ of directed relations , $\overrightarrow{S}$ of Other	84.9
BRCNN	$\overrightarrow{S}$ and $\overleftarrow{S}$ of directed relations, $\overrightarrow{S}$ of Other	**85.4**

Table 3: Comparing different variants of our model.

Bi-RCNN is a variant of BRCNN, which doesn't have the coarse-grained classifier. $\overrightarrow{S}$ and $\overleftarrow{S}$ are shortest dependency paths described in section 2.4. As shown in Table 3, if we inverted the SDP of all relations as input, we observe a performance degradation of 1.2% compared with RCNN. As mentioned in section 3.1, the SemEval-2010 task 8 dataset contains an undirected class Other in addition to 9 directed relations(18 classes). For bidirectional model, it is natural that the inversed Other relation is also in the Other class itself. However, the class Other is used to indicate that relation between two nominals dose not belong to any of the 9 directed classes. Therefore, the class Other is very noisy since it groups many different types of relations with different directions.

On the basis of the analysis above, we only inverse the SDP of directed relations. A significant improvement is observed and Bi-RCNN achieves an F_1-score of 84.9%. This proves bidirectional representations provide more useful information to classify directed relations. We can see that our model still benefits from the coarse-grained classification, which can help our model learn inherent correlation between directed relations with opposite directions. Compared with Bi-RCNN classifying $\overrightarrow{S}$ and $\overleftarrow{S}$ into 19 classes separately, BRCNN also conducts a 10 classes (9 directed relations and Other) classification and improves 0.5% in F_1-score. Beyond the relation classification task, we believe that our bidirectional method is general technique, which is not restricted in a specific dataset and has the potential to benefit other NLP tasks.

4 Related Work

Relation classification is an important topic in NLP. Traditional Methods for relation classification mainly fall into three classes: feature-based, kernel-based and neural network-based.

In feature-based approaches, different types of features are extracted and fed into a classifier. Generally, three types of features are often used. Lexical features concentrate on the entities of interest, e.g., POS. Syntactic features include chunking, parse trees, etc. Semantic features are exemplified by the concept hierarchy, entity class. Kambhatla (2004) used a maximum entropy model for feature combination. Rink and Harabagiu (2010) collected various features, including lexical, syntactic as well as semantic features.

In kernel based methods, similarity between two data samples is measured without explicit feature representation. Bunescu and Mooney (2005) designed a kernel along the shortest dependency path between two entities by observing that the relation strongly relies on SDPs. Wang (2008) provided a systematic analysis of several kernels and showed that relation extraction can benefit from combining convolution kernel and syntactic features. Plank and Moschitti (2013) combined structural information and semantic information in a tree kernel. One potential difficulty of kernel methods is that all data information is completely summarized by the kernel function, and thus designing an effective kernel becomes crucial.

Recently, deep neural networks are playing an important role in this task. Socher et al. (2012) introduced a recursive neural network model that assigns a matrix-vector representation to every node in a parse tree, in order to learn compositional vector representations for sentences of arbitrary syntactic type and length.

Convolutional neural works are widely used in relation classification. Zeng et al. (2014) proposed an approach for relation classification where sentence-level features are learned through a CNN, which has word embedding and position features as its input. In parallel, lexical features were extracted according to given nouns. dos Santos et al. (2015) tackled the relation classification task using a convolutional neural network and proposed a new pairwise ranking loss function, which achieved the state-of-the-art result in SemEval-2010 Task 8.

Yu et al. (2014) proposed a Factor-based Compositional Embedding Model (FCM) by deriving sentence-level and substructure embeddings from word embeddings, utilizing dependency trees and named entities. It achieved slightly higher accuracy on the same dataset than Zeng et al. (2014), but only when syntactic information is used.

Nowadays, many works concentrate on extracting features from the SDP based on neural networks. Xu et al. (2015a) learned robust relation representations from SDP through a CNN, and proposed a straightforward negative sampling strategy to improve the assignment of subjects and objects. Liu et al. (2015) proposed a recursive neural network designed to model the subtrees, and CNN to capture the most important features on the shortest dependency path. Xu et al. (2015b) picked up heterogeneous information along the left and right sub-path of the SDP respectively, leveraging recurrent neural networks with long short term memory units. We propose BRCNN to model the SDP, which can pick up bidirectional information with a combination of LSTM and CNN.

5 Conclusion

In this paper, we proposed a novel bidirectional neural network BRCNN, to improve the performance of relation classification. The BRCNN model, consisting of two RCNNs, learns features along SDP and inversely at the same time. Information of words and dependency relations are used utilizing a two-channel recurrent neural network with LSTM units. The features of dependency units in SDP are extracted by a convolution layer.

We demonstrate the effectiveness of our model by evaluating the model on SemEval-2010 relation classification task. RCNN achieves a better performance at learning features along the shortest dependency path, compared with some common neural networks. A significant improvement is observed when BRCNN is used, outperforming state-of-the-art methods.

6 Acknowledgements

Our work is supported by National Natural Science Foundation of China (No.61370117 & No. 61433015) and Major National Social Science Fund of China (No. 12&ZD227).

References

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.

Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference*, pages 626–634.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. 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. Association for Computational Linguistics*, pages pages 94–99.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Neural computation*, pages 1735–1780.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Joint Conference on Natural Language Processing and the 7th International Joint Conference on Natural Language Processing*, pages 285–290.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Barbara Plank and Alessandro Moschitti. 2013. Embeddings semantic similarity in tree kernels for domain adaption of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1498–1507.

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, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 151–161.

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.

Mengqiu Wang. 2008. A re-examination of dependency path kernels for relation extraction. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 841–846.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of Conference on Empirical Methods in Natural Language Processing,*, pages 1785–1794.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*, pages 95–101.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Mathew D. Zeiler. 2012. An adaptive learning rate method. In *arXiv preprint at arXiv:1212.5701*.

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

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.