Pairwise Relation Classification with Mirror Instances and a Combined Convolutional Neural Network

Jianfei Yu
School of Information Systems
Singapore Management University
jfyu.2014@phdis.smu.edu.sg

Jing Jiang
School of Information Systems
Singapore Management University
jingjiang@smu.edu.sg

Abstract

Relation classification is the task of classifying the semantic relations between entity pairs in text. Observing that existing work has not fully explored using different representations for relation instances, especially in order to better handle the asymmetry of relation types, in this paper, we propose a neural network based method for relation classification that combines the raw sequence and the shortest dependency path representations of relation instances and uses mirror instances to perform pairwise relation classification. We evaluate our proposed models on two widely used datasets: SemEval-2010 Task 8 and ACE-2005. The empirical results show that our combined model together with mirror instances achieves the state-of-the-art results on both datasets.

1 Introduction

Relation classification is a very important task for many Natural Language Processing (NLP) applications including question answering (Yao and Van Durme, 2014), knowledge base population (Socher et al., 2013) and opinion mining (Kobayashi et al., 2007). The goal of relation classification is to automatically identify the semantic relation between a pair of entities in free text. For example, a relation classification system should be able to capture the *Cause-Effect* relation between the entities *pressure* and *burst* in the sentence "The *burst* has been caused by water hammer *pressure*."

Like any classification task, a key research question of relation classification is the identification of a good feature representation for each relation instance. Traditional approaches focus on either combining many manually designed features (Zhou et al., 2005; Jiang and Zhai, 2007; Li and Ji, 2014) or leveraging various kernels to implicitly explore a large feature space (Bunescu and Mooney, 2005; Zhang et al., 2006; Qian et al., 2008; Nguyen et al., 2009), but both approaches suffer from their poor generalization ability on unseen words, and fail to achieve very satisfactory performance (Nguyen et al., 2015). Recently, with the advances of deep learning in NLP, neural networks (NNs) have exhibited their advantages in dealing with unseen words through pre-trained word embeddings and capturing meaningful hidden representations. Different NN architectures, including Convolutional Neural Network (CNN) (Zeng et al., 2014), Recursive Neural Network (ReNN) (Socher et al., 2012) and Recurrent Neural Network (RNN) (Xu et al., 2015b), have been applied to relation classification.

However, most existing NN-based approaches only exploit one of the following structures to represent relation instances: raw word sequences (Zeng et al., 2014; dos Santos et al., 2015), constituency parse trees (Socher et al., 2012; Hashimoto et al., 2013) and dependency parse trees (Xu et al., 2015a; Xu et al., 2015b; Miwa and Bansal, 2016). For the models based on raw sequence, despite maintaining all the information in relation instances, they cannot well handle long-distance relations. For the models based on constituency parse trees, one of the bottlenecks is handling long-distance relations (Ebrahimi and Dou, 2015). For the dependency tree-based models, although they focus on the condensed information captured by the shortest dependency path between the two entities and thus are good at capturing long-distance relations, they lose some supplementary information in the original instance (Liu et al., 2015). Observing that the raw sequence and the dependency path representations highly complement each other, we expect a combination of the two structures to be more effective in capturing long-distance relations without losing any information.

Moreover, another important issue with the feature representation of relation instances is regarding the asymmetry of relation types. Most relation types are asymmetric. Take the *Cause-Effect* relation in the SemEval dataset as an example. *Cause-Effect*(e_1, e_2) indicates that e_1 is the cause and e_2 is the effect. If their roles are reversed, we need to represent the relation as either *Cause-Effect*(e_2, e_1) or *Effect-Cause*(e_1, e_2). Suppose we have K different asymmetric relation types. The current common practice to handle the relation directions is to transform the K+1 class labels (where the +1 is for the *Other* relation, which is symmetric) into $2K$+1 class labels, where each of the K asymmetric relations is expanded into two labels to capture the two directions. For example, from *Cause-Effect*, another label *Effect-Cause* is created. Given any sentence containing two entities, we can always treat the first entity as e_1 and the second entity as e_2. We can then classify their relation into one of the $2K$+1 labels.

Although this approach has been shown to be effective, it neglects the fact that the two class labels corresponding to the same original asymmetric relation are correlated. Take the above-mentioned *burst-pressure* sentence as an example. Most previous methods will treat it as a positive instance for the *Effect-Cause* relation only (because the first entity *burst* in the sentence is the effect). They will not relate the sentence to the *Cause-Effect* relation, although if we treat the second entity *pressure* as e_1, its relation to the first entity *burst* is *Cause-Effect*. We believe that if we represent each relation instance in two ways by swapping the order of the two entities, we can not only implicitly link the pair of relation labels from the same relation but also make a better prediction on a relation instance based on its two representations.

Based on the two observations above regarding the complementary nature of the raw sequence and dependency path representations and the asymmetry of relation types, in this paper, we propose a mirror instance based pairwise relation classification (MI) method using a convolutional neural network that combines raw sequence and dependency path representations. Our MI method creates mirror instances from the original relation instances by swapping the order of the two entities and using the reversed relation label. The method also learns appropriate weights to combine the predictions made on the original instance and the mirror instance for the final prediction.

Evaluation on SemEval-2010 Task 8 and ACE-2005 shows that both mirror instances and combining raw sequence and dependency path representations help improve the performance of relation classification. Our results also show that: (1) by using only half of the negative training instances to generate mirror instances, we can push the F_1 score to 85.0 on SemEval-2010 Task 8 without using any additional manually-crafted, linguistic-driven features; (2) and with only one additional linguistic-driven feature (entity type), we can obtain results competitive with the state-of-the-art results on ACE-2005.

2 Our Proposed Model

In this section, we first formally formulate the task and introduce our notation. We then present our proposed mirror instance method, including the mirror instance generation strategy and our pairwise relation classification framework. Finally, we present our proposed combined CNN models.

2.1 Problem Formulation

A relation instance consists of a sentence with two entities inside tagged as e_1 and e_2. Here e_1 always precedes e_2 in the sentence. Let $\mathcal{R}$ be a set of pre-defined asymmetric relation types, and $\mathcal{S}$ be a set of pre-defined symmetric relations including no relation. A labeled relation instance has a relation label that indicates both the relation existing between the two entities and the direction of the relation. For example, a relation label can be in the form of either $r(e_1, e_2)$ or $r(e_2, e_1)$, where $r \in \mathcal{R} \cup \mathcal{S}$. To

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

Table 1: Relations in SemEval-2010 Task 8.

Relation	
PART-WHOLE	WHOLE-PART
ORG-AFF	AFF-ORG
ART	ART^{-1}
PHYS	PHYS^{-1}
GEN-AFF	AFF-GEN
PER-SOC	
None	

Table 2: Relations in ACE-2005.

make our explanations simpler, we assume that we are always predicting the relation from e_1 to e_2, and therefore for each $r \in \mathcal{R}$, we introduce a reversed relation label rev(r) to capture the cases when relation r is from e_2 to e_1. For example, if r is *Cause-Effect*, then rev(r) is *Effect-Cause*. In total, we have $2K+L$ class labels, where $K = |\mathcal{R}|$ and $L = |\mathcal{S}|$. For the SemEval-2010 Task 8 data, we list all the $2 \times 9 + 1$ class labels in Table 1, where the $+1$ is for the case when there is no relation, denoted by *Other*. For the ACE-2005 data, all the $2 \times 5 + 2$ class labels are listed in Table 2, where $+2$ indicates the symmetric person-social relation and no relation, denoted by *PER-SOC* and *None*.

We further assume that each relation instance has two kinds of word representations. The first is the raw sequence (RSeq) representation, which consists of the sequence of words in the original sentence. The second is the shorted dependency path (SDP) representation, which is the shortest path from e_1 to e_2 in the dependency parse tree of the original sentence.

Let us use $\mathcal{V}$ to denote the vocabulary that contains all unique words in our dataset and $\mathcal{E}$ the set of directed dependency relation labels such as $\xrightarrow{\text{pobj}}$. The RSeq representation of a relation instance contains a sequence of words $(w_1, w_2, \ldots)$ where $w_i \in \mathcal{V}$. In addition, inspired by the work by Zeng et al. (2014), to tag the positions of e_1 and e_2, we assume that each word w_i in the sequence is associated with two position indices p_i and q_i, which indicate the relative distances of w_i from e_1 and e_2, respectively. Take the token "caused" in the previous *burst-pressure* sentence as an example. Since its relative distance to the two entities "burst" and "pressure" are 3 and -4 respectively, its two position indices are 3 and -4. We use $\mathcal{P}$ to denote the set of all possible position indices in our dataset.

The SDP representation can also be regarded as a sequence of tokens $(t_1, t_2, \ldots)$, where each token is either a word or a directed dependency relation, that is, $t_j \in \mathcal{V} \cup \mathcal{E}$. Similar to the RSeq representation, we also use the relative distances of t_j to e_1 and e_2 to indicate the positions of e_1 and e_2, namely c_j and d_j. The left side of the bottom layer of Figure 2 shows the RSeq and the SDP representations of the relation instance "The $[$*burst*$]_{e_1}$ has been caused by water hammer $[$*pressure*$]_{e_2}$."

Formally, we assume that we are given a set of labeled relation instances $\{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, where $y^{(n)}$ is a relation label and $x^{(n)}$ has two kinds of word representations: RSeq$(x^{(n)})$ and SDP$(x^{(n)})$.

2.2 Mirror Instance Method

Our first proposal is a new framework to model each relation instance by a pair of representations. The key idea is to first generate a *mirror* instance from each original relation instance, and then perform joint training and testing by making use of both the original and the mirror instances.

Our method is motivated by the observation that each relation instance can provide us with a pair of examples with opposite directions. For example, "The $[$*burst*$]_{e_1}$ has been caused by water hammer $[$*pressure*$]_{e_2}$." is an original relation instance and is labeled as *Effect-Cause*. If we swap the order of e_1 and e_2, then the resulting mirror instance "The $[$*burst*$]_{e_2}$ has been caused by water hammer $[$*pressure*$]_{e_1}$." should be labeled as *Cause-Effect*. Recall that in standard practice the relation labels *Cause-Effect* and *Effect-Cause* are treated as two unrelated relations. But intuitively these two relation labels are highly

Raw Sequence (RSeq)	**Shortest Dependency Path (SDP)**	**Label**
The $[$*burst*$]_{e_1}$ has been caused by water hammer $[$*pressure*$]_{e_2}$	$[$***burst***$]_{e_1} \xleftarrow{\text{nsubjpass}}$ **caused** $\xrightarrow{\text{prep}}$ **by** $\xrightarrow{\text{pobj}}$ $[$***pressure***$]_{e_2}$	**Effect-Cause**
-1 0 1 2 3 4 5 6 7	**0 1 2 3 4 5 6**	
-8 -7 -6 -5 -4 -3 -2 -1 0	**-6 -5 -4 -3 -2 -1 0**	
The $[$*burst*$]_{e_2}$ has been caused by water hammer $[$*pressure*$]_{e_1}$	$[$***burst***$]_{e_2} \xleftarrow{\text{nsubjpass}}$ **caused** $\xrightarrow{\text{prep}}$ **by** $\xrightarrow{\text{pobj}}$ $[$***pressure***$]_{e_1}$	**Cause-Effect**
-8 -7 -6 -5 -4 -3 -2 -1 0	**-6 -5 -4 -3 -2 -1 0**	
-1 0 1 2 3 4 5 6 7	**0 1 2 3 4 5 6**	

Figure 1: An example of the representations of a mirror instance (in the bottom row) by our method.

related, and should not be independent of each other. By generating a mirror instance from each original instance, we can not only double the number of training data but also implicitly link the two labels r and $\mathrm{rev}(r)$. More importantly, for each testing instance, we can better identify its relation label based on its two representations.

For a relation instance x, let us use $\bar{x}$ to denote its mirror instance that we generate, and for a relation label y, let us use $\mathrm{rev}(y)$ to denote its *mirror label*, which is the reverse of y. Note that if y corresponds to a symmetric relation label, then $\mathrm{rev}(y)$ also corresponds to the same relation label.

Our mirror instance generation idea is inspired by the negative sampling method by Xu et al. (2015a) but our practice is fairly different. In their method, they only create a negative instance for each positive instance by reversing the original SDP, which will cause the expanded training set more biased to negative instances and thus largely reduce the recall of positive instances, whereas in our method, our generated mirror instances are not simply labeled as *Other* (or *None*) but labeled as a reversed relation from the original relation label. As a result, the class distribution of our generated *mirror* training set is almost the same as that of the original training set because of the *mirror* relationship between the original and *mirror* instances. More importantly, they simply expand the original training set with additional negative samples and their training process is the same as that in standard practice, while we propose a different pairwise relation classification framework in which the original and the mirror representations are jointly used for each relation instance.

Mirror Instance Generation

The next question is how we should construct RSeq($\bar{x}$) and SDP($\bar{x}$), the word representations of the mirror instance, such that we can use the same CNN architecture to learn the hidden sentence representations of these mirror instances.

For both SDP($\bar{x}$) and RSeq($\bar{x}$), although we could simply reverse the original representation as was done by Xu et al. (2015a), we feel that this would result in a completely reversed sentence or shortest dependency path that is unnatural. So we adopt the following way of constructing RSeq($\bar{x}$) and SDP($\bar{x}$). We leave the sequence of words untouched. For the position indices, since they are used to indicate the positions of the two entities, we simply swap the two position indices for each word such that the original e_1 now becomes e_2 and the original e_2 now becomes e_1. The bottom row of Figure 1 shows RSeq($\bar{x}$) and SDP($\bar{x}$) for the mirror instance "The $[$*burst*$]_{e_2}$ has been caused by water hammer $[$*pressure*$]_{e_1}$."

Pairwise Relation Classification

Training: Once it is clear how the RSeq and the SDP representations of a mirror instance are constructed, the next challenge is how to train with these pairs of original and mirror instances such that we can make a final prediction for each relation instance.

Essentially, in addition to the original training data $\{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, we now have additional N training instances $\{(\bar{x}^{(n)}, \mathrm{rev}(y^{(n)})\}_{n=1}^{N}$, where $\bar{x}^{(n)}$ is the mirror instance of $x^{(n)}$ and $\mathrm{rev}(\cdot)$ is as defined previously. Moreover, these pairs of original and mirror instances have a one-to-one correspondence relationship, and therefore there should not be any disagreement between their labels.

We therefore design the following loss function to capture two components. The first component is to maximize the log-likelihood of both the original and the mirror instances as follows:

$$J_c = -\sum_{n=1}^{N} \Big(\log p(y^{(n)}|x^{(n)};\mathbf{\Theta}) + \log p(\mathrm{rev}(y^{(n)})|\bar{x}^{(n)};\mathbf{\Theta}') \Big), \qquad (1)$$

where $\mathbf{\Theta}$ and $\mathbf{\Theta}'$ are two sets of parameters respectively in the CNN model of the original instances and the mirror instances, which will be detailed in Section 2.3.

However, the Eqn. (1) above still treats each label separately and cannot link $y^{(n)}$ and $\mathrm{rev}(y^{(n)})$ to capture the relations between $x^{(n)}$ and $\bar{x}^{(n)}$. Consequently, we further construct a one-to-one correspondence relationship between $(x^{(n)}, y^{(n)})$ and $(\bar{x}^{(n)}, \mathrm{rev}(y^{(n)}))$. Intuitively, for $(x^{(n)}, \bar{x}^{(n)})$, the probability of the final label being $y^{(n)}$ should be a weighted combination of the probability of $x^{(n)}$ being $y^{(n)}$ and the probability of $\bar{x}^{(n)}$ being $\mathrm{rev}(y^{(n)})$. We therefore introduce another parameter $\omega \in \mathbb{R}^{2K+L}$ as a weight vector to combine the likelihood of the original and the mirror instances. The loss function is given as follows:

$$J_f = -\sum_{n=1}^{N} \log \Big(\sigma(\omega_{y^{(n)}}) p(y^{(n)}|x^{(n)};\mathbf{\Theta}) + (1-\sigma(\omega_{y^{(n)}})) p(\mathrm{rev}(y^{(n)})|\bar{x}^{(n)};\mathbf{\Theta}') \Big),$$

where $\sigma(\omega_{y^{(n)}}) = \frac{1}{1+e^{-\omega_{y^{(n)}}}}$ is a tradeoff weight between the probability of $x^{(n)}$ being $y^{(n)}$ and the probability of $\bar{x}^{(n)}$ being $\mathrm{rev}(y^{(n)})$.

Finally, we minimize $J_c + J_f$ as our overall objective function. Since the overall objective function consists of two components and each component is related to the other, we propose to jointly optimize them via stochastic gradient descent with shuffled mini-batches, based on the practice by Kim (2014). In our implementation, the learning rate of each parameter is scheduled by Adadelta (Zeiler, 2012) ($\epsilon = 10^{-1}$, $\rho = 0.95$ for ω, and $\epsilon = 10^{-6}$, $\rho = 0.95$ for $\mathbf{\Theta}$ and $\mathbf{\Theta}'$).

Testing: After training with pairs of original and mirror instances, during the testing stage, how to predict the label of a relation instance becomes straightforward. For a test instance x^t, we should again generate its mirror instance $\bar{x}^t$. Thereafter, we can obtain two class distributions by using the trained model, one from x^t and the other from $\bar{x}^t$. Let us use $c(x^t)$ to denote the former and $c(\bar{x}^t)$ the latter. Finally, we can obtain the final class distribution $c(x^t, \bar{x}^t)$ based on $c(x^t)$ and $c(\bar{x}^t)$:

$$c_k(x^t, \bar{x}^t) = \sigma(\omega_k) c_k(x^t) + (1-\sigma(\omega_k)) c_{\mathrm{rev}(k)}(\bar{x}^t), \quad 1 \le k \le 2K+L,$$

where $c_k(x^t)$ and $c_{\mathrm{rev}(k)}(\bar{x}^t)$ represent the probability of x^t having the relation k and $\bar{x}^t$ having the relation $\mathrm{rev}(k)$ respectively, and $c_k(x^t, \bar{x}^t)$ denotes the probability of the pair of relation instances having the relation k. The final predicted label is the relation with the highest probability among $c(x^t, \bar{x}^t)$.

2.3 Our Combined CNN Model

Under the mirror instance based pairwise relation classification (MI) framework, we further target at learning better representations for both the original and the mirror instances. Motivated by the observation that the raw sequence and the dependency path representations highly complement each other, we propose to combine the RSeq and the SDP representations of each relation instance (either the original or the mirror instance) together based on the multi-channel CNN architecture by Kim (2014).

Figure 2 illustrates the whole architecture of the MI framework, which contains two proposed combined CNN models. Each model obtains hidden representations of both the raw sequence and the shortest dependency path of a relation instance and then concatenates them for relation classification. It consists of a lookup layer, a convolution-pooling layer and an MLP layer.

Lookup: The lookup layer maps the input sequences to real-valued embedding vectors. Let $\mathbf{W}_e \in \mathbb{R}^{d_1 \times |\mathcal{V} \cup \mathcal{E}|}$ denote the lookup table for words and directed dependency relations, where each

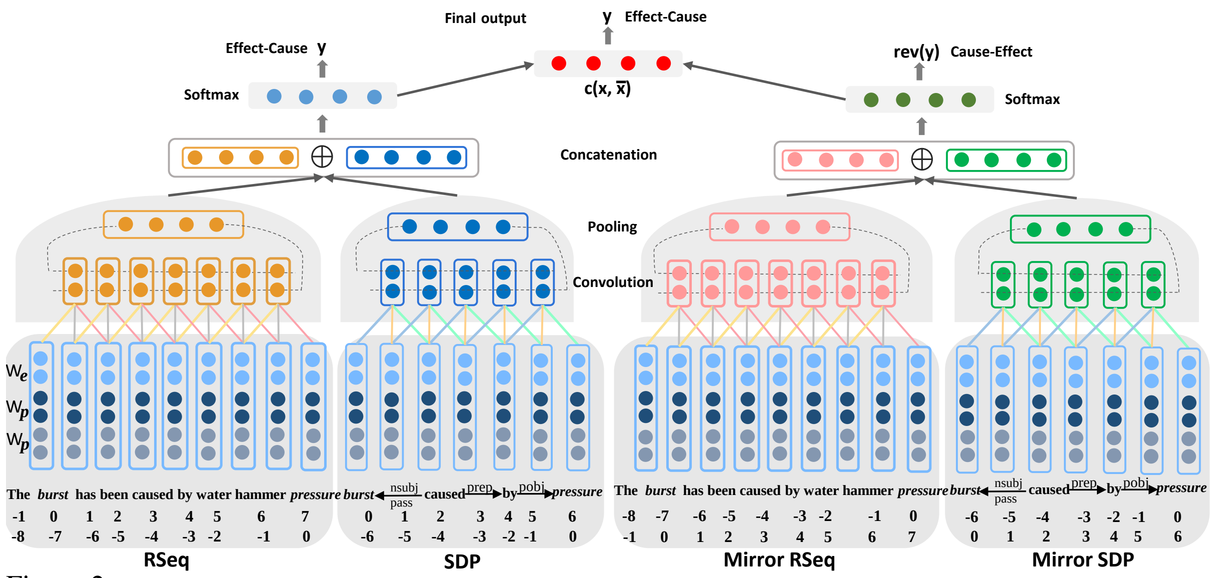

Figure 2: The architecture of the mirror instance based pairwise relation classification (MI) framework. The left and right components correspond to two combined CNN models (*Comb*) respectively for original instances and mirror instances.

column is a d_1-dimensional embedding vector for either a word in $\mathcal{V}$ or a dependency relation in $\mathcal{E}$. Let $\mathbf{W}_p \in \mathbb{R}^{d_2 \times |\mathcal{P}|}$ denote another lookup table for position indices, where each column is a d_2-dimensional embedding vector for a position index. Note that this position embedding idea is borrowed from the work by Zeng et al. (2014), and thus word representations of the raw sequence in our combined model is the same as theirs. After applying the lookup layer, both the RSeq and the SDP representations are transformed into a sequence of $(d_1 + 2d_2)$-dimensional vectors.

Convolution-Pooling: Two separate CNNs are used to process the RSeq representation and the SDP representation, and their mechanisms are the same. For each CNN, at position i of the original sequence, the embedding vectors inside a window of size n centered at i are concatenated into a new vector, which we refer to as $\mathbf{z}_i \in \mathbb{R}^d$. A convolution operation is then performed by applying a filter $\mathbf{F} \in \mathbb{R}^{h \times d}$ on $\mathbf{z}_i$ to produce a hidden vector $\mathbf{h}_i = g(\mathbf{F}\mathbf{z}_i + \mathbf{b})$, where $\mathbf{b} \in \mathbb{R}^h$ is a bias vector and g is a element-wise non-linear transformation function. Note that we pad the original sequence in front and at the back to ensure that at each position i we have n vectors to be combined into $\mathbf{h}_i$. After the convolution operation is applied to the whole sequence, we obtain $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots]$, and we apply a max-over-time pooling operator to take the maximum value of each row of $\mathbf{H}$ to obtain an overall hidden vector $\mathbf{h}^*$, which encodes the information from the entire sequence. Let $\mathbf{h}_r^*$ denote this hidden vector derived from RSeq and $\mathbf{h}_s^*$ the hidden vector derived from SDP.

MLP: The top layer of our model is a multilayer perceptron (MLP) with a softmax layer at the end to predict a $(2K + L)$-class distribution. This means the objective function for training our model is $J(\mathbf{\Theta}) = -\sum_{n=1}^{N} \log p(y^{(n)}|x^{(n)}; \mathbf{\Theta})$, where $\mathbf{\Theta}$ is the set of all model parameters including $\mathbf{W}_e$, $\mathbf{W}_p$, $\mathbf{F}$, $\mathbf{b}$ and the weights in the multilayer perceptron, $y^{(n)}$ is the true relation label for relation instance $x^{(n)}$, and $p(y^{(n)}|x^{(n)}; \mathbf{\Theta})$ is the probability of assigning $y^{(n)}$ to $x^{(n)}$ based on the softmax layer. As discussed in Section 2.2, in our implementation, the gradients are computed via back propagation.

3 Experiments

3.1 Dataset and Evaluation Metric

To evaluate our proposed method, we conduct our experiments on the SemEval-2010 Task 8 dataset and the English portion of the ACE-2005 dataset.

SemEval-2010 Task 8: This dataset contains 10,717 relation instances, including 8000 instances for training and 2717 for testing. Following Kim (2014), we randomly choose 10%, i.e., 800 of the training

Method	Prec	Rec	F$_1$
RSeq	81.11	84.72	82.78
RSeq+MI	**81.23**	**85.42**	**83.22***
SDP	80.57	83.54	82.01
SDP+MI	**80.98**	**83.89**	**82.36***

Table 3: Evaluation of our mirror instance method. * indicates that our method significantly improves the corresponding baseline with $p < 0.05$ based on McNemar's test.

Method	Prec	Rec	F$_1$
RSeq	81.11	84.72	82.78
SDP	80.57	83.54	82.01
Comb	**81.27**	**85.33**	**83.20***

Table 4: Comparison of our combined model with two baseline models using either *RSeq* or *SDP* representation.

Method	Prec	Rec	F$_1$
Comb	81.27	85.33	83.20
Comb+MI	**82.07**	**86.63**	**84.23***
Comb+RMI	***82.34***	***87.76***	***84.96****

Table 5: Evaluation of our combined CNN model together with the mirror instance method. RMI stands for reduced mirror instances.

instances as the development set. Following all previous work, we use the macro-averaged F_1 score to evaluate our model based on the SemEval-2010 Task 8 official scorer.

ACE-2005: This dataset consists of 6 domains: broadcast news (*bn*), newswire (*nw*), broadcast conversation (*bc*), telephone conversation (*cts*), weblogs (*wl*) and usenet (*un*). Following some previous work (Plank and Moschitti, 2013; Nguyen et al., 2015; Gormley et al., 2015), we consider a domain adaptation setting for coarse-grained relation extraction. Specifically, we take the union of *bn* and *nw* as the training set, half of *bc* as the development set, and the remainder (i.e., *cts* and *wl* as well as the other half of *bc*) as the test set. Following Plank and Moschitti (2013), we use the micro-averaged F_1 score to evaluate our model.

3.2 Experiment Settings

We use the pre-trained word embeddings from *word2vec*[1] to initialize the lookup table $\mathbf{W}_e$, and set the dimension d_1 to 300. For unknown words and directed dependency labels, we randomly initialize their 300-dimensional embedding vectors. We also randomly initialized the other lookup table of the position embeddings $\mathbf{W}_p$, and set the dimension d_2 to 50. Note that in our preliminary experiments for ACE-2005, we found that the performance without considering the entity types of the two entity mention heads is very limited. Hence, for ACE-2005, we also randomly initialize another lookup table of the entity type embeddings, whose dimension is set to 50, and represent each token by concatenating its word embedding, position embedding and entity embedding.

We want to compare our combined CNN model with models that use either RSeq or SDP alone, so we consider three experiment settings: *SDP* refers to a CNN model that uses only SDP representation of a relation instance, *RSeq* refers to a CNN model that uses only the RSeq, and *Comb* refers to our combined model. For each setting, we use the development set to tune the window size n and the dimension of the hidden states h. In a previous study by Nguyen and Grishman (2015), it was found that using multiple window sizes in CNN can bring significant improvements for the RSeq representation. We therefore also experiment with combining multiple window sizes for *RSeq* and *SDP*. In the end, we find that for *RSeq*, the optimal setting is to use a combination of windows with sizes 2, 3, 4 and 5 and to set h to 150. For *SDP*, the optimal setting is to use a single window of size 5 and to set h to 400. For *Comb*, we use the same window sizes and hidden sizes h as *RSeq* and *SDP*.

For the other parameters in $\mathbf{\Theta}$ and $\mathbf{\Theta}'$, we adopt the settings reported by Nguyen and Grishman (2015). That is, the non-linear transformation function g is $\tanh$, the mini-batch size is 50, the dropout rate α equals 0.5, and the hyperparameter for the l_2 norms is set to be 3.

3.3 Evaluation of our Proposed Approach

In this section, we evaluate the different components of our method.

Effect of the Mirror Instance Method

To evaluate the effect of the mirror instances, first, we apply the mirror instance method on top of the two baseline methods *RSeq* and *SDP*, and show the results on SemEval-2010 in Table 3 and the results

[1]`https://code.google.com/p/word2vec/`

Method	dev set			bc			cts			wl			avg
	Prec	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**F_1**
RSeq	73.5	52.7	61.4	70.3	52.4	60.1	65.8	45.9	54.1	57.7	44.1	50.0	54.7
RSeq+MI	69.1	59.1	**63.7***	65.7	59.2	**62.3***	67.7	44.9	54.0	59.5	46.9	**52.4***	**56.2**
SDP	67.6	49.3	57.0	59.4	45.3	51.4	55.4	37.7	44.9	48.8	36.6	41.8	46.0
SDP+MI	66.5	51.7	**58.2***	57.6	48.6	**52.7***	53.3	39.3	**45.3**	45.8	37.4	41.2	**46.4**

Table 6: Evaluation of our mirror instance method on ACE-2005.

Method	dev set			bc			cts			wl			avg
	Prec	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**F_1**
RSeq	73.5	52.7	61.4	70.3	52.4	60.1	65.8	45.9	54.1	57.7	44.1	50.0	54.7
SDP	67.6	49.3	57.0	59.4	45.3	51.4	55.4	37.7	44.9	48.8	36.6	41.8	46.0
Comb	72.4	57.8	**64.3***	70.6	57.5	**63.4***	62.5	49.7	**55.4** *	60.4	49.5	**54.4***	**57.7**

Table 7: Evaluation of our combined CNN model on ACE-2005.

on ACE-2005 in Table 6. We can see that with the help of the mirror instances, both *RSeq* and *SDP* can improve their performance in most cases, and the improvements are statistically significant. This indicates the usefulness of the mirror instances generated by our method.

The Combined CNN Model

To check the effect of combining RSeq and SDP representations, in Table 4 and Table 7, we compare *Comb* with *SDP* and *RSeq* on SemEval-2010 and ACE-2005 respectively. We can observe that *Comb* outperforms both *SDP* and *RSeq* on two datasets and the improvements are statistically significant. We can also see that the precision of *Comb* and that of the other two models are relatively close, especially on SemEval-2010, and the advantage of *Comb* is mainly from its recall. It suggests that the RSeq and the SDP representations complement each other and therefore can work better when combined.

The Combined CNN Model together with the Mirror Instance Method

We then apply our mirror instance method on top of *Comb*. In Table 5 and the top two rows of Table 9, we can observe that our mirror instance method (*Comb*+MI) can significantly improve the F_1 score of *Comb*, especially making high improvements in recall, which further verifies the usefulness of our mirror instance method.

Since the goal of our relation classification task is to improve the F_1 score for the positive relation types excluding the label *Other*, we further investigate the impact of reducing the number of mirror instances generated from the *Other* relation instances, i.e., the negative relation instances on SemEval-2010. By tuning the percentage of negative mirror instances to reduce, we achieve the best performance when reducing 50% of the negative mirror instances. We refer to this method as *Comb*+RMI and show its performance in Table 5 in the last row. We can see that it achieves a F_1 score of 84.96.

3.4 Comparison with the State of the Art

In this section, we compare our proposed method with all recently published results for SemEval-2010 Task 8 and ACE-2005.

SemEval-2010 Task 8: Since most existing studies have used additional hand-crafted linguistic features (AF) to help the classification task, we show two different F_1 scores, one with AF and one without in Table 8. It is easy to observe that without AF, Vu et al. (2016) obtained the best F_1 score of 84.9 by combining CNN and RNN models via a voting strategy; with AF, Xu et al. (2015b) achieved the best result with negative sampling and 8000 negative examples from the New York Times (NYT) dataset.

Method	Additional Features (AF)	F_1	
		with AF	without AF
SVM (Rink and Harabagiu, 2010)	POS, prefixes, morphological, WordNet, Levin classed, ProBank, FrameNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2	-
CNN (Zeng et al., 2014)	words around entities, WordNet	82.7	78.9
DepNN (Liu et al., 2015)	NER	83.6	82.8
Hybrid(FCM+Feat) (Gormley et al., 2015)	NER	83.7	-
SDP-LSTM (Xu et al., 2015b)	POS,Wordnet	83.7	82.4
DepLNN+NS (Xu et al., 2015a)	Samples from NYT, WordNet	85.6	84.0
CR-CNN (dos Santos et al., 2015)	-	-	84.1^{+}
ER-CNN + RNN (Vu et al., 2016)	-	-	84.9^{+}
SpTree (Miwa and Bansal, 2016)	Wordnet	85.5	84.5
Comb+RMI	Wordnet, NER	**85.7**	**85.0**

Table 8: Comparisons with state-of-the-arts results on SemEval-2010. $^{+}$ indicates using a special ranking-based objective function.

Method	**dev set**			**bc**			**cts**			**wl**		
	Prec	**Rec**	$\mathbf{F_1}$	**Prec**	**Rec**	$\mathbf{F_1}$	**Prec**	**Rec**	$\mathbf{F_1}$	**Prec**	**Rec**	$\mathbf{F_1}$
Comb	72.4	57.8	64.3	70.6	57.5	63.4	62.5	49.7	55.4	60.4	49.5	54.4
Comb+MI	70.9	60.4	**65.2***	66.2	63.6	**64.9***	65.1	51.5	***57.5****	57.1	52.4	**54.7**
The State-of-the-art Systems												
FCM	-	-	-	66.6	57.9	61.9	65.6	44.3	52.9	57.8	44.6	50.4
Hybrid(FCM+Feat)	-	-	-	74.4	55.3	63.5	74.5	45.0	56.1	65.6	47.6	55.2
Hybrid(CNN+RNN+Feat)	69.3	66.3	***67.8***	65.8	66.5	***66.1***	63.6	51.7	57.0	56.4	57.2	***56.8***

Table 9: Evaluation of our combined CNN model together with the mirror instance method on ACE-2005. The results of the state-of-the-art systems are taken from Nguyen and Grishman (2016).

We can also see that without utilizing any AF, our *Comb*+RMI method can push the F_1 score to the state-of-the-art, 85.0. Furthermore, we also consider adding two kinds of lexical features to our model, namely, Named Entity type (NER) and WordNet hypernyms. We first obtain the NER features of all words and Wordnet hypernyms of the two entities using the tool developed by Ciaramita and Altun (2006). Then, we represent each token by concatenating its word embedding, position embedding and entity embedding. Finally, following the practice by Zeng et al. (2014), we also concatenate the Wordnet hypernyms of the two entities with the combined hidden vector. As we can see from the last line of Table 8, our method can achieve the state-of-the-art F_1 score, 85.7.

ACE-2005: In Table 9, it is easy for us to observe that on all three test domains, our proposed *Comb*+MI method can outperform the state-of-the-art single system FCM with a large margin, which combines traditional linguistic features with learned word embeddings by a log-bilinear model. In addition, we can also find that even in comparison with a competitive hybrid model, which integrates FCM and a traditional feature-based method, *Comb*+MI can still achieve slightly better performance on the *bc* and *cts* domains, and similar performance on the *wl* domain.

Recently, Nguyen and Grishman (2016) proposed an ensemble method by first combining CNN and RNN via a stacking strategy and then integrating it with a traditional feature-based method in a hyrid model. Although our result is slightly lower than Hybrid(CNN+RNN+Feat) on average, we believe that our model can be further improved with such an ensemble strategy, which we leave to our future work.

4 Related Work

Traditional work on relation classification can be categorized into feature-based methods and kernel-based methods. The former relies on a large number of human-designed features (Zhou et al., 2005; Jiang and Zhai, 2007; Li and Ji, 2014) while the latter leverages various kernels to implicitly explore a much larger feature space (Bunescu and Mooney, 2005; Nguyen et al., 2009). However, both methods suffer from error propagation problems and poor generalization abilities on unseen words. The most popular method to solve the two limitations is based on neural networks (NNs), which have been shown successful in extracting meaningful features and generalizing on unseen words for many NLP tasks (Kim, 2014). For relation classification, Socher et al. (2012) proposed a recursive matrix-vector model based on constituency parse trees. Zeng et al. (2014) and dos Santos et al. (2015) respectively proposed a standard and a ranking-based CNN model based on the raw word sequences. More recently, Xu et al. (2015b) and Miwa and Bansal (2016) respectively proposed a multi-channel sequential LSTM model and a bidirectional tree-LSTM model on the shortest dependency path for relation classification.

Although all these models have been shown to be effective, all of them only focus on learning a single representation for each relation instance. Different from all previous methods, we first design a strategy to generate a mirror instance from each original relation instance and then propose a pairwise relation classification framework to learn a pair of representations for each relation instance.

On the other hand, most existing NN-based approaches for relation classification are either based on the shortest dependency path or the raw sequence, although these two representations may complement each other. In this work, we propose to combine them together based on the multi-channel CNN architecture (Kim, 2014), aiming to capture long-distance relations without losing any information.

5 Conclusions

In this paper, we first proposed a mirror instance method to learn a pair of representations for each instance, which basically includes a mirror instance generation strategy and a pairwise relation classification framework. Based on this, we further proposed a combined CNN model based on both the RSeq and the SDP representations of relation instances. Our experimental results demonstrate that our mirror instance method can improve the baseline models and our combined model without mirror instances, and our combined CNN model is more effective than models only using the RSeq or the SDP representation of relation instances. Finally, with the help of some lexical features, our combined CNN model together with the mirror instance method achieves the state-of-the-art result on SemEval-2010 and highly competitive results on ACE-2005.

Acknowledgment

We would like to thank the reviewers for their valuable comments. This research is supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

References

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731. Association for Computational Linguistics, October.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602. Association for Computational Linguistics.

Cicero 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 Natural Language Processing (Volume 1: Long Papers)*, pages 626–634. Association for Computational Linguistics, July.

Javid Ebrahimi and Dejing Dou. 2015. Chain based rnn for relation classification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1244–1249. Association for Computational Linguistics, May–June.

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1774–1784. Association for Computational Linguistics, September.

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. Association for Computational Linguistics, October.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 113–120. Association for Computational Linguistics, April.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, October.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *EMNLP-CoNLL*, volume 7, pages 1065–1074. Citeseer.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland, June. 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 Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 285–290. Association for Computational Linguistics, July.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48. Association for Computational Linguistics, June.

Thien Huu Nguyen and Ralph Grishman. 2016. Combining neural networks and log-linear models to improve relation extraction. *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence*.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1378–1387. Association for Computational Linguistics, August.

Thien Huu Nguyen, Barbara Plank, and Ralph Grishman. 2015. Semantic representations for domain adaptation: A case study on the tree kernel-based method for relation extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 635–644, Beijing, China, July. Association for Computational Linguistics.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1498–1507. Association for Computational Linguistics.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 697–704. Association for Computational Linguistics.

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, July.

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. Association for Computational Linguistics, July.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems*.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 534–539, San Diego, California, June. Association for Computational Linguistics.

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. Association for Computational Linguistics, September.

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 the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794. Association for Computational Linguistics, September.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland, June. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 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. Dublin City University and Association for Computational Linguistics, August.

Min Zhang, Jie Zhang, Jian Su, and GuoDong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 825–832. Association for Computational Linguistics, July.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434. Association for Computational Linguistics, June.