Using Cost-Sensitive Ranking Loss to Improve Distant Supervised Relation Extraction

Daojian Zeng, Junxin Zeng, Yuan Dai

Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410004, P. R. China
zengdj@csust.edu.cn
zeng_jx@stu.csust.edu.cn daiy@stu.csust.edu.cn

Abstract. Recently, many researchers have concentrated on using neural networks to learn features for Distant Supervised Relation Extraction (DSRE). However, these approaches generally employ a softmax classifier with cross-entropy loss, and bring the noise of artificial class *NA* into classification process. Moreover, the class imbalance problem is serious in the automatically labeled data, and results in poor classification rates on minor classes in traditional approaches.
In this work, we exploit cost-sensitive ranking loss to improve DSRE. It first uses a Piecewise Convolutional Neural Network (PCNN) to embed the semantics of sentences. Then the features are fed into a classifier which takes into account both the ranking loss and cost-sensitive. Experiments show that our method is effective and performs better than state-of-the-art methods.

1 Introduction

There has been many methods proposed for relation extraction. In these methods, the supervised paradigm has been shown to be effective and yield relatively high performance [8, 19]. However, a large labeled training data is often required for this task, and manually annotating large labeled training data is a time-consuming and labor intensive task.

To address the shortcomings of supervised paradigm, distantly supervised [11] paradigm is proposed to automatically generate training data. Traditional methods have typically applied supervised models to elaborately handcrafted features when obtained the labeled data through distant supervision [11, 14, 5, 15]. These features are often derived from preexisting Natural Language Processing (NLP) tools, thus inevitably have errors. With the recent revival of interest in neural networks, many researchers have investigated the possibility of using neural networks to automatically learn features for relation classification [18, 17, 7, 9]. The neural networks based methods achieve substantial improvements in the task, however, they still have the following deficiencies.

First, it brings the noise of artificial class *NA* into the classification process. Previous methods usually employ a Convolutional Neural Network (CNN) to

Relations	Number of Samples
NA	158513
/location/location/contains	2966
/people/person/place_lived	792
/people/person/nationality	711
/business/person/company	498
/people/person/place_of_birth	483
/people/deceased_person/place_of_death	263
/location/neighborhood/neighborhood_of	177
/business/company/founders	87

Table 1. The distribution of the data that is generated through distant supervision strategy.

embed the semantics of sentences. The learned vectors are subsequently fed into a softmax classifier and the whole network is learned to minimize a categorical cross-entropy loss function. Unfortunately, the artificial class *NA* is very noisy since it groups many different infrequent relation types. Table 1 shows the distribution of the training data that is generated through distant supervision strategy. The samples corresponding to *NA* account for vast majority of the illustrated relations. Thus, the noise in *NA* cannot be ignored.

Second, it has class imbalance problem in the automatically labeled training data, and shows poor classification rates. From Table 1, we can observe that the class imbalance problem is indeed serious. For example, the number of samples corresponding to */location/location/contains* is about 34 times that of */business/company/founders*. It tends to be biased toward the major classes when using the training data generated through distant supervision.

In this paper, we exploit a cost-sensitive ranking loss function to address the two problems described above. We use a PCNN to automatically learn relevant features and incorporates multi-instance To address the noise of artificial class *NA*, the PCNN is followed by a ranking-based classifier instead of a softmax classifier. In the classifier, we adopt a pairwise ranking loss function and do not learn the class embedding for *NA*. Moreover, we incorporate cost-sensitive in the loss function to alleviate the class imbalance problem. In this work, we give different margins to different classes in order to achieve cost-sensitive classification. The experimental results show that our model achieves significant and consistent improvements as compared with the baseline systems.

2 Related Work

Relation extraction is one of the most important topics in NLP. Supervised approaches are the most commonly used methods for relation extraction and yield relatively high performance [2, 16, 19]. In the supervised paradigm, relation extraction is considered to be a multi-class classification problem and may suffer from a lack of labeled data for training. To address this issue, [11] adopts Freebase

to perform distant supervision. As described in Section 1, the algorithm for training data generation is sometimes faced with the wrong label problem. To address this shortcoming, [14, 5, 15] develop the relaxed distant supervision assumption for multi-instance learning. [12] utilize relation definitions and Wikipedia documents to improve their systems

The methods mentioned above have been shown to be effective for DSRE. However, their performance depends strongly on the quality of the designed features. Recently, many researchers attempt to use deep neural networks in DSRE without handcrafted features. [18] adopts CNNs to embed the semantics of the sentences. Moreover, [4] proposes a pairwise ranking loss function in the CNNs to reduce the impact of artificial class. These methods build classifier based on sentence-level annotated data, which cannot be directly applied for DSRE since multiple sentences corresponding to a fact may be achieved in the data generating procedure. Therefore, [17] incorporates multi-instance learning with neural network model, which can build relation extractor based on distant supervision data. Although the method achieves significant improvement in relation extraction, it only selects the most likely sentence for each entity pair in their multi-instance learning paradigm. To address this issue, [9] proposes sentence level attention over multiple instances in order to utilize all informative sentences. [7] employs cross-sentence max-pooling to select features across different instances, and then aggregates the most significant features for each entity pair.

The aforementioned works, especially neural networks, have greatly promoted the development of relation extraction. However, these works do not pay attention to the noise of artificial class and the class imbalance problem, which are unfortunately very common in DSRE. In this work, we use cost-sensitive ranking loss to address these problems in DSRE. Different from [4], we incorporate cost sensitive in the loss function and our approach involves the problem of instance selection as well.

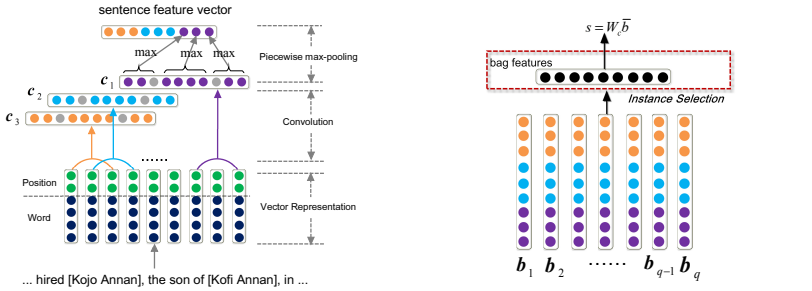
3 Methodology

Figure 1 shows the neural network architecture used in this work. It consists of two parts: PCNNs Module and Ranking Based Classifier Module. We describe these parts in details below.

3.1 PCNNs Module

This module is used to extract feature vector of an instance (sentence) in a bag. PCNNs differs from traditional CNN by devising piecewise max pooling layer instead of the single max pooling layer.

Vector Representation The inputs of our network are raw word tokens. When using neural networks, we typically transform word tokens into low-dimensional vectors. In this paper, the "word token" refers to word and entity. In the following, we do not distinguish them and call them "word". In our method, each input



(a) PCNNs Module (b) Ranking Based Classifier Module

Fig. 1. The architecture used in this work.

word token is transformed into a vector by looking up pre-trained word embeddings. Moreover, we use Position Features (PFs) [18, 17] to specify entity pairs, which are also transformed into vectors by looking up position embeddings.

Word Embeddings Word embeddings are distributed representations of words that map each word in a text to a 'k'-dimensional real-valued vector. They have recently been shown to capture both semantic and syntactic information about words very well, setting performance records in several word similarity tasks [10, 13]. Using word embeddings that have been trained a priori has become common practice for enhancing many other NLP tasks [6]. In the past years, many methods for training word embeddings have been proposed [1, 3, 10]. We employ the method [10] to train word embeddings and denote it by $\mathbf{E}$.

Position Embeddings [17] has shown the importance of PFs in relation extraction. Similar to their works, we use PFs to specify entity pairs. A PF is defined as the combination of the relative distances from the current word to entity e_1 and entity e_2. We randomly initialize two position embedding matrices $\mathbf{PF}_i (i = 1, 2)$ (for e_1 and e_2), and transform the relative distances into vectors by looking them up.

We concatenate the word representation and position representation as the input of the network (shown in Figure 1(a)). Assume that the size of word representation is k_w and that of position representation is k_d, then the size of a word vector is $k = k_w + 2k_d$.

Convolution Assume that $\mathbf{A} = (a_{ij})_{m\times n}$ and $\mathbf{B} = (b_{ij})_{m\times n}$, then the convolution of $\mathbf{A}$ and $\mathbf{B}$ is defined as $\mathbf{A} \otimes \mathbf{B} = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}b_{ij}$.

We denote the input sentence by $S = \{s_1, s_2, \cdots, s_{|S|}\}$ where s_i is the i-th word, and use $\boldsymbol{s}_i \in \mathbb{R}^k$ to represent its vector. We use $\mathbf{S}_{i:j}$ to represent the matrix concatenated by sequence $[\boldsymbol{s}_i : \boldsymbol{s}_{i+1} : \cdots : \boldsymbol{s}_j]$ ($[\boldsymbol{x}_1 : \boldsymbol{x}_2]$ denotes the horizontal concatenation of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$). We denote the length of filter by w

(Figure 1(a) shows an example of $w = 3$), then the weight matrix of the filter is $\mathbf{W} \in \mathbb{R}^{w \times k}$. Then the convolution operation between the filter and sentence S results in another vector $\boldsymbol{c} \in \mathbb{R}^{|S|-w+1}$:

$$c_j = \mathbf{W} \otimes \mathbf{S}_{(j-w+1):j} \tag{1}$$

where $1 \leq j \leq |S| - w + 1$.

In experiments, we use $n(n > 1)$ filters (or feature maps) to capture different features of an instance. Therefore, we also need n weight matrices $\mathbf{W}_c = \{\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_n\}$, so that all the convolution operations can be expressed by

$$c_{ij} = \mathbf{W}_i \otimes \mathbf{S}_{(j-w+1):j} \tag{2}$$

where $1 \leq i \leq n$ and $1 \leq j \leq |S| - w + 1$. Through the convolution layer, we obtain the results vectors $\mathbf{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_n\}$.

Piecewise Max Pooling In order to capture the structural information and fine-grained features, PCNNs divides an instance into three segments according to the given entity pair (two entities cut the sentence into three parts) and do max-pooling operation on each segment. For the result vector $\boldsymbol{c}_i$ of convolution operations, it can be divided into three parts $\boldsymbol{c}_i = \{\boldsymbol{c}_{i,1}, \boldsymbol{c}_{i,2}, \boldsymbol{c}_{i,3}\}$. Then piecewise max-pooling procedure is $p_{ij} = \max(\boldsymbol{c}_{i,j})$, where $1 \leq i \leq n$ and $j = 1, 2, 3$. After that, we can concatenate all the vectors $\boldsymbol{p}_i = [p_{i,1}, p_{i,2}, p_{i,3}](i = 1, 2, \cdots, n)$ to obtain vector $\boldsymbol{p} \in \mathbb{R}^{3n \times 1}$. Figure 1(a) displays an example of $n = 3$, in which the gray circles are the positions of entities. Finally, we compute the feature vector $\boldsymbol{b}_S = \tanh(\boldsymbol{p})$ for sentence S.

3.2 Ranking Based Classifier Module

The PCNNs module extracts the feature vector for each sentence. The ranking based classifier module is responsible for selecting the most appropriate instance in a bag and predicting the most likely relation.

Classifier To compute the score for each relation, the feature vector of each instance is fed into a ranking based classifier. Given the distributed vector representation of an instance $\boldsymbol{b}$, the network computes the score for a class label t_i by using the dot product:

$$s_{t_i} = \mathbf{w}_{t_i} \boldsymbol{b} \tag{3}$$

where $\mathbf{w}_{t_i} \in \mathbb{R}^{1 \times 3n}$ is the class embedding for class label t_i. All the class embeddings $\mathbf{w}_{t_i}$ $(i = 1, \cdots, T)$ constitute the class embedding matrix $\mathbf{W}_T \in \mathbb{R}^{T \times 3n}$ whose rows encode the distributed vector representations of the different class labels. T is equal to the number of possible relation types for the relation extraction system. Note that the number of dimensions in each class embedding must be equal to the size of the distributed vector representation of the input bag $3n$. The class embedding matrix $\mathbf{W}_T$ is a parameter to be learned by the network.

Instance Selection Distant supervised relation extraction suffers from wrong label problem [14]. The core problem that needs to be solved in the multi-instance learning is to get the corresponding bag feature vector from all the instance feature vectors in the bag. In fact, the problem is the instance selection strategy. We employ an instance selection strategy borrowed from [17]. Different from [17], we randomly select an instance from the bag with *NA* label since our model do not give score for *NA* class (see Section 3.2). In addition, we choose the instance which has the highest score for the bag label except for *NA*. The scores are computed using Equation (3). Therefore, our instance selection strategy will not be disturbed by the noise in *NA*. Assume that there is a bag $B_i = \{b_1^i, b_2^i, \cdots, b_{q_i}^i\}$ that contains q_i instances with feature vectors $\{\boldsymbol{b}_1^i, \boldsymbol{b}_2^i, \cdots, \boldsymbol{b}_{q_i}^i\}$ and the bag label is r_i ($r_i \neq NA$). The j-th instance b_j^i is selected and the j is constrained as follows:

$$j = \arg\max\{s_{r_i}^1, s_{r_i}^2, \cdots, s_{r_i}^{q_i}\} \quad 1 \leq j \leq q_i \tag{4}$$

where $s_{r_i}^j = \mathbf{w}_{r_i} \boldsymbol{b}_j^i$ $(1 \leq j \leq q_i)$ is computed using Equation (3).

Cost-sensitive Ranking Loss As mentioned in section 1, the cross-entropy loss brings the noise of artificial class into the classification process. This phenomenon is mainly due to the noise of artificial class *NA*. To address this shortcoming, we propose a pairwise ranking loss instead of cross-entropy which is often used for softmax classifier.

In our model, the network can be stated as a tuple $\theta = (\mathbf{E}, \mathbf{PF}_1, \mathbf{PF}_2, \mathbf{W}_c, \mathbf{W}_T)$. Assume that there are N bags in training set $\{B_1, B_2, \cdots, B_N\}$, and their labels are relations $\{r_1, r_2, \cdots, r_N\}$. After the instance selection, we get a representative instance and its corressponding feature vector is considered as the bag feature vector $\bar{\boldsymbol{b}}$. The input for each iteration round is a bag feature vector and the class label. In the pairwise ranking, the loss function is defined on the basis of pairs of objects whose labels are different. We can get the ranking loss function through selecting a class label that is different from the input one. In this work, we choose the negative class with the highest score among all incorrect classes. For example, when the i-th bag with label $r_i = t_j$ is fed into the network, we select a negative class t_k which obtains highest score except class t_j. Using Equation (3), we can get the classification scores of the i-th bag $s_{t_j}^i$ and $s_{t_k}^i$ for class t_j and t_k, respectively. The ranking loss function is define as follows:

$$\begin{aligned} \mathcal{L} = \sum_{i=1}^{N} \{ & \log(1 + \exp(\lambda(m^+ - s_{t_j}^i))) \\ & + \log(1 + \exp(\lambda(m^- + s_{t_k}^i)))\} \end{aligned} \tag{5}$$

where $t_k \neq t_j$ and $j, k \in \{1, 2, \cdots, T\}$. λ is a scaling factor that magnifies the difference between the scores. m^+ and m^- are the margin for correct and incorrect class, respectively.

DSRE confronts with another problem is the class imbalance. To alleviate this shortcoming, we incorporate cost-sensitive in the loss function and different

margins are given to different classes. The margins are computed as follows:

$$m_{t_i} = \gamma \times \frac{\log(\#t_i)}{\sum_{j=1}^{T} \log(\#t_j)} \tag{6}$$

where $\gamma > 0$ is a constant term. $\#t_j$ represents the number of samples corresponding to relation t_j. Substituting m_{t_i} into Equation (5), the cost-sensitive ranking loss function is shown as follows:

$$\begin{aligned} \mathcal{L} = \sum_{i=1}^{N} \{ & \log(1 + \exp(\lambda(m_{t_j} - s^i_{t_j}))) \\ & + \log(1 + \exp(\lambda(m_{t_k} + s^i_{t_k})))\} \end{aligned} \tag{7}$$

From the cost-sensitive ranking loss function, we can observe that the first term in the right side of Equation (7) decreases as the score s_{t_j} increases and the second term decreases as the score s_{t_k} decreases. The cost-sensitive ranking loss function aims to give scores greater than m_{t_j} for the correct class t_j and smaller than m_{t_k} for incorrect class t_k. When the minor classes are incorrectly classified, our model gives them more penalties than the major classes. We use the back-propagation algorithm and stochastic gradient descent (SGD) to minimize the loss function with respect to θ.

It is very difficult to learn patterns for the artificial class *NA*. Therefore softmax classifier often brings noise into the classification process of the natural classes. We can avoid explicitly leaning patterns for the artificial class when using ranking loss function. We omit the artificial class *NA* by setting the first term in the right side of Equation (7) to zero and do not learn the class embedding for *NA*. Thus, our model does not give score for the artificial class *NA* and the noise in *NA* is alleviated. At prediction time, an instance is classified as *NA* only if all actual classes have negative scores. A bag is positively labeled if and only if the output of the network on at least one of its instances is assigned a positive label and we choose the class which has the highest score.

4 Experiments

In this section, we first introduce the dataset and evaluation metrics, then test several variants via cross-validation to determine the parameters used in our experiments, finally show the experimental results and analysis.

4.1 Dataset and Evaluation Metrics

We evaluate our method on a widely used dataset[1] that was developed by [14] and has also been used by [5, 15, 17]. This dataset was generated by aligning

[1] `http://iesl.cs.umass.edu/riedel/ecml/`

Parameters	*Value*
Window size	$w = 3$
Feature maps	$n = 230$
Word dimension	$d_w = 50$
Position dimension	$d_p = 5$
Batch size	$b_s = 50$
Adadelta parameter	$\rho = 0.95, \varepsilon = 1e^{-6}$
Constant term	$\lambda = 2, \gamma = 50$

Table 2. Parameters used in our experiments.

Freebase relations with the NYT corpus, with sentences from the years 2005-2006 used as the training corpus and sentences from 2007 used as the testing corpus.

Following the previous work [11], we evaluate our approach using held-out evaluation. The held-out evaluation compares the extracted relation instances against Freebase relation data.

4.2 Experimental Settings

In this work, we use the Skip-gram model (word2vec)[2] to train the word embeddings on the NYT corpus. The tokens are concatenated using the ## operator when the entity has multiple word tokens. Position features are randomly initialized with uniform distribution between $[-1, 1]$. For convenience of comparing with baseline methods, the PCNNs module uses the same parameter settings as [17]. We use L2 regularization with regularization parameter $\beta = 0.001$ and set the max number of iteration to 200. We tune the proposed model using three-fold validation to study the effects of two parameters: the constant terms λ and γ used in the loss function. We use a grid search to determine the optimal parameters and manually specify subsets of the parameter spaces: $\lambda \in \{1, 2, \cdots, 10\}$ and $\gamma \in \{10, 20, \cdots, 100\}$. Table 2 shows all parameters used in the experiments.

4.3 Baselines

We select three previous works that use handcrafted features as well as the CNN-based methods as baselines. *Mintz* is proposed by [11] which extracts features from all instances; *MultiR* is a multi-instance learning method proposed by [5]; *MIML* is a multi-instance multi-labels method proposed by [15]; *PCNNs+MIL* is the method proposed by [17], which incorporates multi-instance learning with PCNNs to extract bag features; *CrossMax* is proposed by [7], which exploits PCNNs and cross-sentence max-pooling to select features across different instances.

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

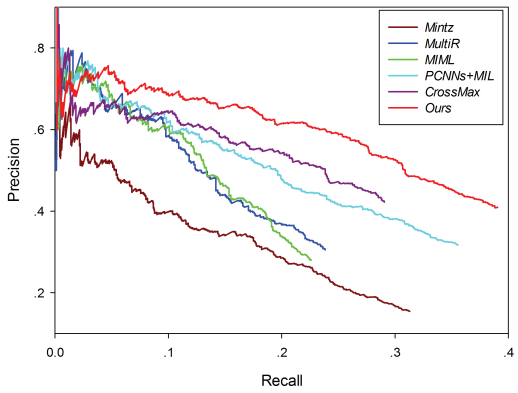


Fig. 2. Performance comparison of proposed method and baseline methods.

4.4 Comparison with Baseline Methods

In this section, we show the experimental results and comparisons with baseline methods. In the following experiments, we use *Ours* to refer to the proposed model that use cost-sensitive ranking loss.

The held-out evaluation provides an approximate measure of precision without requiring costly human evaluation. Half of the Freebase relations are used for testing. The relation instances discovered from the test articles are automatically compared with those in Freebase.

For convenience of comparing with baseline methods, the prediction results are sorted by the classification scores and a precision-recall curve is created for the positive classes. Figure 2 shows the precision-recall curves of our approach and all the baselines. We can observe that our model outperforms all the baseline systems and improves the results significantly. It is worth emphasizing that the best of all baseline methods achieves a recall level of 36%. In contrast, our model is much better than the previous approach and enhances the recall to approximately 39%. The significant improvement can be contributed to the magic of ranking based classifier. The ranking based classifier use ranking loss which avoids explicitly learning the patterns for *NA*. Thus, our model will not trend to classify the samples as *NA* compared to sofmax classifier and recalls more positive samples.

Furthermore, our model achieves a large improvement in precision especially at higher recall levels. From Figure 2, we can see that our model achieves precision of 40% when recall is 39%. In contrast, when *PCNNS* and *CrossMax* achieve such precision, the recalls are decreased to approximately 24% and 29%, respectively. Thus, our approach is advantageous from the point of view of precision. This improvement can be contributed to the cost-sensitive. The class imbalance problem is alleviated through cost-sensitive, and the precision is improved.

Also note that our model does not show advantages in the precision when the recall is very low. This phenomenon is mainly due to the fact that when

Relations	*Ours-CS*			*Ours*		
	P	R	F1	P	R	F1
/location/location/contains	35.45	43.44	39.04	**36.67**	**43.50**	**39.79**
/people/person/place_lived	12.87	18.67	15.24	**16.31**	**17.86**	**17.05**
/people/person/nationality	**47.79**	24.53	32.42	46.54	**25.32**	**32.80**
/business/person/company	35.41	**52.48**	**42.29**	**36.48**	49.87	42.14
/people/person/place_of_birth	11.45	16.82	13.62	**15.66**	**17.77**	**16.65**
/people/deceased_person/place_of_death	26.31	22.22	24.09	**26.43**	**24.59**	**25.47**
/location/neighborhood/neighborhood_of	37.50	29.34	32.92	**38.55**	**31.81**	**34.86**
/business/company/founders	47.05	**28.76**	**35.70**	**47.23**	28.42	35.49

Table 3. Precision, recall and F1 score of some relations. *Ours-CS* means that do not use cost sensitive.

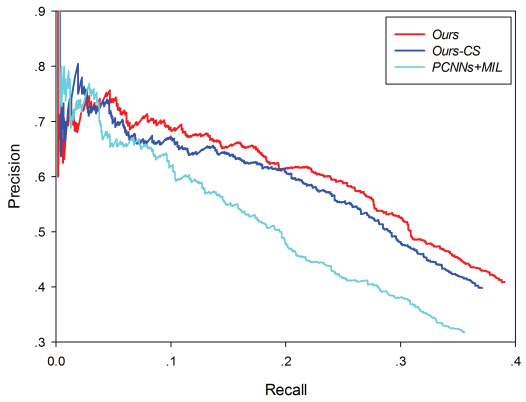


Fig. 3. Effects of cost-sensitive. *Ours-CS* means that do not use cost sensitive.

using held-out evaluation, some examples with high classification score are false negatives and are actually true relation instances. Therefore, we can conclude that our model outperforms all the baseline systems and improves the results significantly in terms of both precision and recall.

4.5 Effects of Cost-sensitive and Ranking Loss

In order to validate the effects of cost-sensitive, we compute the confusion matrix and analyze the detail of some relations in Table 3. From Table 3, we can see that: (1) It achieves better results in the majority of relations when using cost-sensitive; (2) The F1 score is lower in */people/person/place_lived* and */people/person/place_of_birth*, mainly due to the fact that these two relations are more difficult to separate from each other. Nonetheless, the cost-sensitive helps to improve the performance in this case.

The precision-recall curves with and without cost-sensitive are illustrated in Figure 3, from which we can also observe that it brings better performance

when adding cost-sensitive in the loss function. After removing the cost-sensitive ranking loss, our model degrades to *PCNNs+MIL*. In order to further validate the effects of ranking loss, the *PCNNs+MIL* result is illustrated in Figure 3. As we expected, ranking loss take effects and *Ours-CS* can get better performance compared with *PCNNS*. The superiority of our approach indicates that using cost-sensitive ranking loss function can effectively improve DSRE.

5 Conclusions

In this paper, we exploit cost sensitive ranking loss to improve DSRE. We pay attention to the noise of artificial class *NA* and the class imbalance problem in DSRE. Experimental results show that the proposed approach offers significant improvements over comparable methods. The noise of artificial class and the class imbalance problem can be effectively addressed by using cost-sensitive ranking loss. In the future, we would like to further investigate how different loss functions and different cost-sensitive strategies influence performance.

6 Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.61602059), Hunan Provincial Natural Science Foundation of China (No. 2017JJ3334), the Research Foundation of Education Bureau of Hunan Province, China (No. 16C0045), and the Open Project Program of the National Laboratory of Pattern Recognition (NLPR). We thank the anonymous reviewers for their insightful comments.

References

1. Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.
2. Razvan Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In Y. Weiss, B. Schoelkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems, Vol. 18: Proceedings of the 2005 Conference (NIPS)*, 2006.
3. Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.
4. Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL*, 2015.
5. Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 541–550, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

6. Fei Huang, Arun Ahuja, Doug Downey, Yi Yang, Yuhong Guo, and Alexander Yates. Learning representations for weakly supervised natural language processing tasks. *Comput. Linguist.*, 40(1):85–120, March 2014.
7. Xiaotian Jiang, Quan Wang, Peng Li, and Bin Wang. Relation extraction with multi-instance multi-label convolutional neural networks. In *Proceedings of COLING*, pages 1471–1480, 2016.
8. Nanda Kambhatla. 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*, ACLdemo '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.
9. Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*, pages 2124–2133, Stroudsburg, PA, USA, 2016. Association for Computational Linguistics.
10. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.
11. Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
12. Truc-Vien T. Nguyen and Alessandro Moschitti. End-to-end relation extraction using distant supervision from external semantic repositories. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 277–282, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.
13. Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*, pages 1746–1751, 2014.
14. Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of ECML PKDD*, pages 148–163, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
15. Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 455–465, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.
16. Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106, March 2003.
17. Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, pages 17–21, Stroudsburg, PA, USA, 2015. Association for Computational Linguistics.
18. Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Zhao. Zhao, Jun. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344, 2014.
19. GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 427–434, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.