An Adversarial Training Framework for Relation Classification

Wenpeng Liu[1,2], Yanan Cao[1(✉)], Cong Cao[1], Yanbing Liu[1], Yue Hu[1], and Li Guo[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
`{liuwenpeng,caoyanan,caocong,liuyanbing,huyue,guoli}@iie.ac.cn`
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Relation classification is one of the most important topics in Natural Language Processing (NLP) which could help mining structured facts from text and constructing knowledge graph. Although deep neural network models have achieved improved performance in this task, the state-of-the-art methods still suffer from the scarce training data and the overfitting problem. In order to solve this problem, we adopt the adversarial training framework to improve the robustness and generalization of the relation classifier. In this paper, we construct a bidirectional recurrent neural network as the relation classifier, and append word-level attention to the input sentence. Our model is an end-to-end framework without the use of any features derived from pre-trained NLP tools. In experiments, our model achieved higher F1-score and better robustness than comparative methods.

Keywords: Relation classification · Deep learning · Adversarial training Attention mechanism

1 Introduction

Relation Classification is the process of recognizing the semantic relations between pairs of *nominals*. It is a crucial component in natural language processing and could be defined as follows: given a sentence S with the annotated pairs of nominals e_1 and e_2, we aim to identify the relations between e_1 and e_2. For example: "The $[\text{singer}]_{e1}$, who performed three of the nominated songs, also caused a $[\text{commotion}]_{e2}$ on the red carpet." Our goal is to find out the relation of marked entities *singer* and *commotion*, which is obviously recognized as *Cause-Effect* (e_1, e_2) relation in this demonstration.

Traditional relation classifiers generally focused on features representation or kernel-based approaches which rely on full-fledged NLP tools, such as POS tagging, dependency parsing and semantic analysis [13, 14]. Although these approaches are able to exploit the symbolic structures in sentences, they still suffer from the weakness of using handcrafted features. In recently years, deep learning models which extract features automatically, have achieved big improvements on this task. Commonly used models include convolutional neural network (CNN), recurrent neural network (RNN) and other complex hybrid networks [7, 8]. In the most recent past, some researchers combined features representation with neural network models to utilize more characteristics, such as the shortest dependency path [2].

© Springer International Publishing AG, part of Springer Nature 2018
Y. Shi et al. (Eds.): ICCS 2018, LNCS 10861, pp. 194–205, 2018.
https://doi.org/10.1007/978-3-319-93701-4_15

Although deep neural network architectures have achieved state-of-the-art performance, to train an optimized model relies on a large amount of labeled data, otherwise it will lead to overfitting. Due to the high cost of manually tagging samples, in many specific tasks, labeled data is scarce and may not fully sustain the training of a deep supervised learning model. For example, in relation classification task, the standard dataset just contains 10,717 annotated sentences. To prevent overfitting, strategies such as dropout [16] and adding random noise [17, 18] have been proposed, but the effectiveness is limited.

In order to address this problem, we innovatively adopt the *adversarial training* framework for classifying the inter-relations between nominals. We generate *adversarial examples* [11, 12] for labeled data by making small perturbations on word embeddings of the input, which significantly increase the loss incurred by our model. Then, we regularize our classifier using adversarial training technique, i.e. training the model to correctly classify both unmodified examples and perturbed ones. This strategy not only improves the robustness to adversarial examples, but also promotes generalization performance for original examples. In this work, we construct a bidirectional LSTM model as a relation classifier. Beyond the basic model, we use a word-level attention mechanism [6] on the input sentence to capture its most important semantic information. This framework is an end-to-end one without using extra knowledge and NLP systems.

In experiments, we run our model and ten typical comparative methods on the SemEval-2010 Task 8 dataset [13]. Our model achieved an F1-score of 88.7% and outperformed other methods in the literature, which demonstrates the effectiveness of adversarial training.

2 Related Work

Traditional methods for relation classification are mainly based on features representation or kernel-based approaches which rely on a mature NLP tools, such as POS tagging, dependency parsing and semantic analysis. [21] propose a shortest path dependency kernel for relation classification, the main idea of which is that the relation strongly relies on the dependency path between two given entities. Besides considering the structural information, [20] introduce semantic information into kernel methods. In these approaches, the use of features extracted by NLP tools results in cascaded error. On the other hand, handcrafted features of data have bad reusability for other tasks.

In order to extract features automatically, recent researches focus on utilizing deep learning models for this task and have achieved big improvements. [9] proposed convolutional neural networks (CNNs), which uses word embedding and position as input. [5, 7] observed that recurrent neural networks (RNNs) with long-short term memory (LSTMs) could improve addressing this problem. Recently, [6] proposed CNNs with two levels of attention for this task in order to better discern patterns in heterogeneous contexts, which achieved the best effect. What is more, some researchers combined features representation with neural network in order to utilize more linguistic information. The typical operations are neural architecture which leverages the shortest dependency path-based CNNs [2], and the SDP-LSTM model [5]. Existing studies revealed that, deep and rich neural network architectures are more capable of information integration and abstraction, while the annotated data maybe not sufficient for the further promotion of performance.

Adversarial Training was originally introduced by image classification [12]. Then it is adapted to text classification and extended to some semi-supervised tasks by [10]. Predecessors' work demonstrated that the learned input with adversarial training have improved in quality, which solved overfitting problem to some extent. Having a similar intuition, [18] added random noise to the input and hidden layer during training, however the effectiveness of randomly adding mechanism is limited. As another strategy for prevent overfitting, dropout [16] is a regularization method widely used for many tasks. We especially conducted an experiment to make a comparison among adversarial training and these methods.

3 Our Model

Given a sentence *s* with a pair of entities e_1 and e_2 annotated, the task of relation classification is to identify the semantic relation between and e_1 and e_2 in accordance with a set of predefined relation types (all types will be displayed in Sect. 4). Figure 1 shows the overall architectures of our adversarial neural relation classification (ANRC).

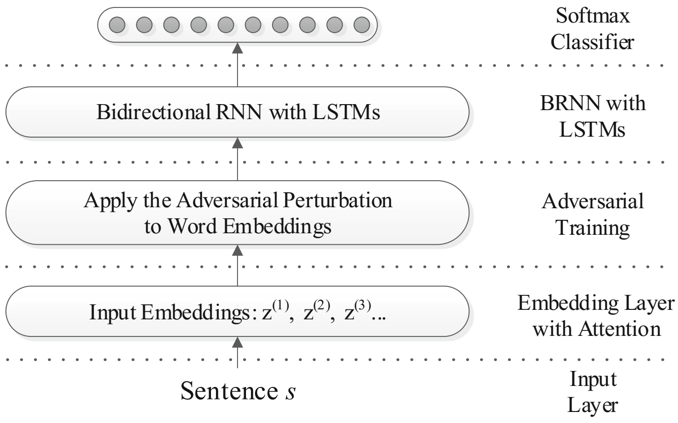
Softmax Classifier
Bidirectional RNN with LSTMs
BRNN with LSTMs
Apply the Adversarial Perturbation to Word Embeddings
Adversarial Training
Input Embeddings: $z^{(1)}$, $z^{(2)}$, $z^{(3)}$...
Embedding Layer with Attention
Sentence *s*
Input Layer

Fig. 1. Overall architecture for adversarial neural relation classification

The input of architecture is encoded using vector representations including word embedding, context and positional embedding. What's more, word-level attention could be used to capture the relevance of words with respect to the target entities. In order to enhance the robustness of model, adversarial examples are leveraged in input embeddings. After that, bidirectional recurrent neural network is used to capture information in different levels of abstraction, and the last layer is a softmax classifier to optimize classification results.

3.1 Input Representation with Word-Level Attention

Given a sentence s, each word w_i is converted into a real-valued vector r^{w_i}. The position embedding of w_i is mapped to a vector of dimension d^{wpe}, tagged as WPE (word position embeddings) proposed by [9]. Consequently, the word embedding and the word position embedding of each word w_1 are concatenated to form the input, $emb_x = \left\{\left[r^{w_1}, wpe^{w_1}\right], \left[r^{w2}, wpe^{w2}\right], \ldots, \left[r^{w_N}, wpe^{w_N}\right]\right\}$. Afterwards, the convolutional operation is applied to each window of size k of successive windows in $emb_x = \{r^{w_1}, r^{w_2}, \ldots, r^{w_N},\}$, ultimately, we define vector z_n as the concatenation of a sequence of k word embedding, centralized in the n-th word:

$$Z_n = (r^{w_{n-(k-1)/2}}, \cdots, r^{w_{n+(k-1)/2}})^T \tag{1}$$

Word-Level Attention. Attention mechanism makes the neural network look back to the key parts of the source text when it is trying to predict the next token of a sequence. Attentive neural networks have been applied successfully in sequence-to-sequence learning tasks. In order to fully capture the relationships and interest of specific words with the target nominals, we design a model to automatically learn this relevance for relation classification like [6].

Contextual Relevance Matrices. Take notice of the example in Fig. 2, we can easily observe that the non-entity word "cause" is of great significance to determine the relation of entity pair. For the sake of characterizing the contextual correlations and connections between entity mention e_j and non-entity word w_i, we leverage two diagonal attention matrix A^j with value $A^j_{i,i} = f(e_j, w_i)$, which is computed as the inner product between embeddings of the entity e_j and word w_i respectively. Based on the diagonal attention matrixes, the relativeness of the i-th word with respect to j-th entity ($j \in \{1, 2\}$) could be calculated as Eq. (1):

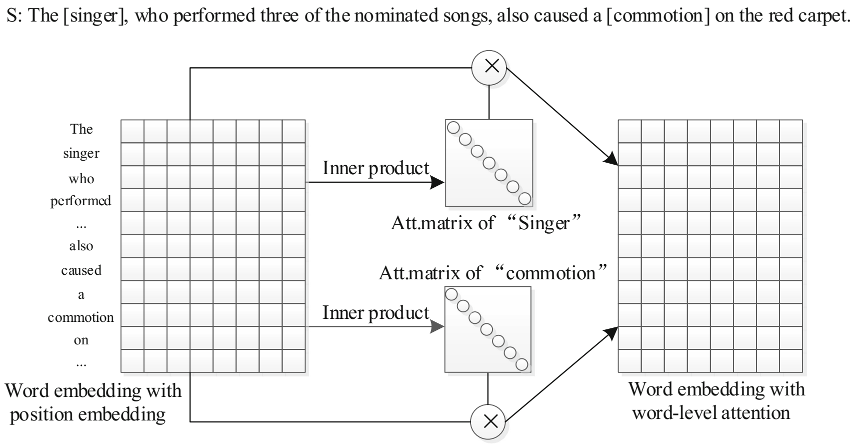
S: The [singer], who performed three of the nominated songs, also caused a [commotion] on the red carpet.
The
singer
who
performed
...
also
caused
a
commotion
on
...
Word embedding with position embedding
Inner product
Att.matrix of "Singer"
Att.matrix of "commotion"
Inner product
Word embedding with word-level attention

Fig. 2. Word-level attention on input

$$\alpha_i^j = \frac{\exp\left(A_{i,i}^j\right)}{\sum_{i'=1}^n \exp\left(A_{i',i'}^j\right)} \tag{2}$$

Input Attention Composition. Next, we combine the two relevance factors α_i^1 and α_i^2 with compositional word embedding z_n above in for recognizing the relation via a simple average algorithm as:

$$r_i = z_i \cdot \frac{\alpha_i^1 + \alpha_i^2}{2} \tag{3}$$

Finally, we've got the final output of word-level attention mechanism, a matrix $R = [r_1, r_2, \ldots, r_n]$ where n is the sentence length, regarded as input vectors feed into neural network we construct.

3.2 Bi-LSTM Network for Classification

Bi-LSTM Network. As a text classification model, we use a LSTM-based neural network model which is used in the state-of-the-art works [1, 7] and the experimental results show its effectiveness for this problem. Beyond the basic model, we adopt in our method a variant introduced by [15]. The LSTM-based recurrent neural network consists of four components: an input gate, a forget gate, an output gate, and a memory cell .

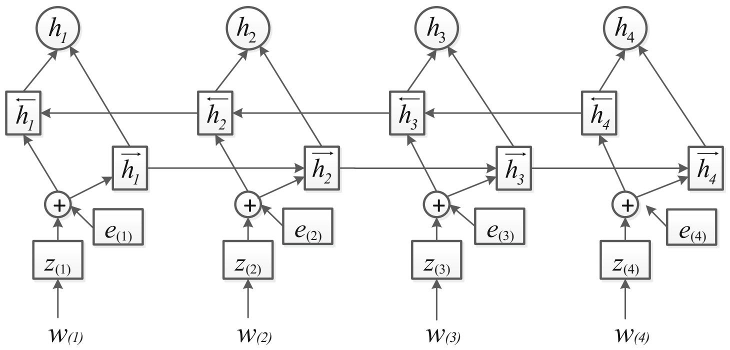

Fig. 3. The model of Bi-LSTMs and perturbed embeddings

We employ the bidirectional recurrent neural network in this part so as to better capture the textual information from both ends of the sentences in view of the fact that the standard RNN is a biased model, where the later inputs are more dominant than the earlier inputs.

Softmax Layer. The softmax layer is a commonly used classifier, which can be regarded as a generalization of multivariate classifier from binary Logistic Regression (LR) one. For this part, we use it to predict the label y from a discrete set of classes Y for a sentence. We denote s as the input sentence and θ as the parameters of a classifier. The output of Bi-LSTM

h is the input of the classifier (Eq. (4)). Simply taking the summation over the log probabilities of all those labels yields the final loss function as Eq. (5).

$$p(y|s;\theta) = \mathrm{softmax}(W_y * h + b_y) \tag{4}$$

$$L(s;\theta) = -\sum_{i=1}^{|Y|} \log P\left(y_i|s;\theta\right) \tag{5}$$

3.3 Adversarial Training

Adversarial examples are generated by making small perturbations to the input, which is designed to significantly increase the loss incurred by a machine learning model. And *adversarial training* is a way of regularizing supervised learning algorithms to improves robustness to small, approximately word case perturbations. It's a process of training a model to correctly classify unmodified examples and adversarial examples.

As shown in Fig. 3, we apply the adversarial perturbation to word embeddings, rather than directly to the input, which is similar to [10]. We denote the concatenation of a sequence of word embedding vectors $[z^{(1)}, z^{(2)}, \ldots, z^{(T)}]$ as s'. Then we define the adversarial perturbation e_{adv} on s' as Eq. (6). Here e is a perturbation on the input and $\hat{\theta}$ denotes a fixed copy of the current value of θ.

$$e_{adv} = arg \min_{\|e\| \le \epsilon} -L\left(s' + e;\hat{\theta}\right) \tag{6}$$

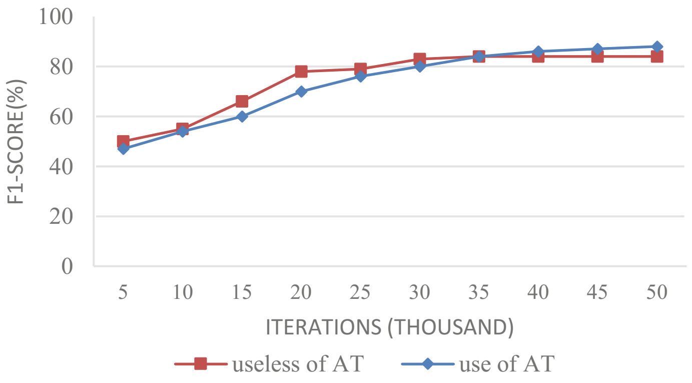

Fig. 4. Training progress of ANRC and ANRC minus AT across iterations

When applied to a classifier, adversarial training adds e_{adv} to the cost as Eq. (7) instead of Eq. (5), where N in Eq. (7) denotes the number of labeled examples. The adversarial training is carried out to minimize the negative log-likelihood plus L_{adv} with stochastic gradient descent.

$$L_{adv}(s';\theta) = -\frac{1}{N}\sum_{n=1}^{N}\log p(y_n|s'_n + e_{adv,n};\theta) \tag{7}$$

At each step of training, we identify the worst perturbations e_{adv} against the current model $p\left(y|s';\widehat{\theta}\right)$, and train the model to be robust to such perturbations through minimizing Eq. (7) with respect to θ. However, Eq. (6) is computationally intractable for neural nets. Inspired by [11], we approximate this value by linearizing $L(s';\theta)$ around s as Eq. (8).

$$e_{adv} = \frac{\epsilon g}{\|g\|},\ where\ g = \nabla_s L(s';\hat{\theta}) \tag{8}$$

4 Experiments and Results

4.1 Datasets

Our experiments are conducted on SemEval-2010 Task 8 dataset, which is widely used for relation classification [13]. The dataset contains 10,717 annotated examples, including 8,000 sentences for training and 2,717 for testing. The relationships between nominals in the corpus are classified into 10 categories, which are list as below. We adopt the official evaluation metric to evaluate our systems, which is based on macro-averaged F1-score for the nine actual relations (Table 1).

Table 1. 9 relationships and examples in our dataset

Relation	Example
Cause-effect	"The <e1>burst</e1> has been caused by water hammer<e2>pressure</e2>"
Component-whole	The ride-on <e1>boat</e1> <e2>tiller</e2> was developed by engineers Arnold S. Juliano and Dr. Eulito U. Bautista
Content-container	This cut blue and white striped cotton <e1>dress</e1> with red bands on the bodice was in a <e2>trunk</e2> of vintage Barbie clothing
Entity-origin	One basic trick involves a spectator choosing a <e1>card</e1> from the<e2>deck</e2> and returning it
Entity-destination	Both his <e1>feet</e1> have been moving into the <e2>ball</e2>
Message-topic	This <e1>love</e1> of nature's gift has been reflected in <e2>artworks</e2> dating back more than a thousand years
Member-collection	In the corner there are several gate captains and a <e1>legion</e1> of Wu <e2>crossbowmen</e2>
Instrument-agency	A <e1>thief</e1> who tried to steal the truck broke the igenition with <e2>screwdriver</e2>
Product-producer	A <e1>factory</e1> for <e2>cars</e2> and spareparts was built in Russia
Other	The following information appeared in the <e1>notes</e1> to consolidated financial <e2>statements</e2> of some corporate annual reports

4.2 Comparative Methods

To evaluate the effectiveness of our model, we compare its performance with notable traditional machine learning approaches and deep learning models including CNN, RNN and other neural network architectures. The comparative methods are introduced in the following.

- **Traditional machine learning algorithms:** As a traditional handcrafted-feature based classification, [19] fed extracted features from many external corpora to an SVM classifier and achieved 82.2% F1 score.
- **RNN based models:** MV-RNN is a recursive neural network build on the constituency tree and achieved a comparable performance with SVM [22]. SDP-LSTM is a type of gated recurrent neural network, and it is the first attempt to use LSTM in this task and it raised the F1-score to 83.7% [5].
- **CNN based models:** [9] construct a CNN on the word sequence and integrated word position embedding, make a breakthrough on the task. CR-CNN extended the basic CNN by replacing the common softmax cost function with a ranking-based cost function [3], and achieved an F1-score of 84.1%. Using a simple negative sampling method, depLCNN + NS introduced additional samples from other corpora like the NYT dataset. And this strategy effectively improved the performance to 85.6% F1-score [4]. Att-Pooling-CNN appended multi-level attention to the basic CNN model, and have achieved the state-of-the-art F1-score in relation classification task [6].
- **RNN combined with CNN:** DepNN is a convolutional neural network with a recursive neural network designed to model the subtrees, and achieve an F1-score of 83.6% [2].

4.3 Experimental Setup

We utilize the word embeddings with 200 dimensions released by Stanford[1]. For model parameters, we set the dimension of the entity position feature vector as 20. We use Adam optimizer with batch size 64, an initial learning rate of 0.001 and a 0.99 learning rate exponential decay factor at each training step. The word window size on the convolutional layer is fixed to 3. We also leverage dropout method to training the neural network with 0.5 dropout ratio. For adversarial training, we empirically choose "ϵ" = 0.02. We trained for 50,000 steps for each method in contrast experiments.

We run all experiments using TensorFlow on two Tesla V100 GPUs. Our model took about 8 min per epoch on average.

4.4 Results Analysis

Comparation with Other Models. Table 2 presents the best effect achieved by our adversarial-training based model (ANRC) and comparative methods. We observe that our model achieves an F1-score of 88.7%, outperforming the state-of-the-art models.

[1] https://nlp.stanford.edu/projects/glove/.

Table 2. Results of our model and comparative methods

Model	F1 (%)
Methods of traditional classifier	
SVM [19]	82.2
Neural networks with dependency features	
MVRNN [22]	82.4
Hybrid FCM [24]	83.4
SDP-LSTM [5]	83.7
DRNNs [1]	85.8
SPTree [23]	84.5
Neural works (End-to-end)	
CNN+Softmax [9]	82.7
CR-CNN [3]	84.1
DepNN [2]	83.6
depLCNN+NS [4]	85.6
Att-Pooling-CNN [6]	88.0
Our architecture	
ANRC	88.7

From the results in Table 2, we can also find that, in the end-to-end frameworks the CNN architectures have achieved better performance than RNN ones. Besides, the employment of negative sampling in depLCNN+NS promote the F1-score to more than 85%. And the attention mechanism introduced in the Att-Pooling-CNN model significantly improved the effectiveness of relation classification. Although we use a Bi-LSTM as the basic classification model, there is still some improvement in the performance, which proved the effectiveness of adversarial training framework.

Robustness of Adversarial Training. In order to test the robustness of our model, we delete half of the training data, and evaluate the models' precision on training data and test data respectively. All using the Bi-LSTM model with attention as the relation classifier, we adopt three different strategies to prevent overfitting: adversarial training plus dropout, adding random noise plus dropout, and just using dropout. Comparative results are shown in Table 3. Although the Adversarial Training+Dropout method has a little precision loss on training data, it achieves an acceptable precision on test data which prominently outperforms other strategies. It demonstrates that training with adversarial perturbations well alleviated the overfitting in the case of scarce training data. Meanwhile, our model has stronger robustness to small, approximately word case perturbations.

Table 3. F1-score in the case of halving training data

Strategy for reducing overfitting	Precision (training data)	Precision (test data)
Dropout	83.1%	59.6%
Random noise+dropout	82.3%	66.4%
Adversarial training+dropout	81.0%	75.5%

Convergence of Adversarial Training. We compare the convergence behavior of our method using adversarial training to that of the baseline Bi-LSTM model with attention. We plot the performance of each iteration of these two models in Fig. 4. From this figure, we find that training with adversarial examples converges more slowly while the final F1 score is higher. It enlightens us that, we could pre-trained the model without adversarial training to faster the process.

5 Conclusion and the Future Work

In this paper, we proposed an adversarial training framework for relation classification, named ANRC, to improve the performance and robustness of relation classification. Experimental results demonstrate that, training with adversarial perturbations outperformed the method with random perturbations and dropout in term of reducing overfitting. And, our model using a Bi-LSTM relation classifier with word-level attention outperforms previous models. In the future work, we will construct various relation classifier models and apply the adversarial training framework on other tasks.

Acknowledgement. This work was supported by the National Key Research and Development program of China (No. 2016YFB0801300), the National Natural Science Foundation of China grants (No. 61602466).

References

1. Xu, Y., Jia, R., Mou, L., Li, G., Chen, Y., Lu, Y., Jin, Z.: Improved relation classification by deep recurrent neural networks with data augmentation. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 1461–1470 (2016)
2. Liu, Y., Wei, F., Li, S., Ji, H., Zhou, M., Wang, H.: A dependency-based neural network for relation classification. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (2015)
3. dos Santos, C., Xiang, B., Zhou, B.: Classifying relations by ranking with convolutional neural networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (2015)
4. Xu, K., Feng, Y., Huang, S., Zhao, D.: Semantic relation classification via convolutional neural networks with simple negative sampling. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 536–540 (2015)
5. Xu, Y., Mou, L., Li, G., Chen, Y., Peng, H., Jin, Z.: 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, pp. 1785–1794 (2015)
6. Wang, L., Cao, Z., de Melo, G., Liu, Z.: Relation classification via multi-level attention CNNs. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers, vol. 1, pp. 1298–1307 (2016)
7. Cai, R., Zhang, X., Wang, H.: Bidirectional recurrent convolutional neural network for relation classification. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers, vol. 1, pp. 756–765 (2016)

8. Zeng, D., Liu, K., Chen, Y., Zhao, J.: Distant supervision for relation extraction via piecewise convolutional neural networks. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1753–1762 (2015)
9. Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J.: Relation classification via convolutional deep neural network. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pp. 2335–2344 (2014)
10. Miyato, T., Dai, A.M., Goodfellow, I.: Adversarial training methods for semi-supervised text classification. arXiv preprint arXiv:1605.07725 (2016)
11. Goodfellow, I.J., Shlens, J., Szegedy, C., Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICML, pp. 1–10 (2015)
12. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
13. Hendrickx, I., Kim, S.N., Kozareva, Z., Nakov, P., Ó Séaghdha, D., Padó, S., Pennacchiotti, M., Romano, L., Szpakowicz, S.: 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, pp. 94–99. Association for Computational Linguistics (2009)
14. Kambhatla, N.: 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, p. 22. Association for Computational Linguistics (2004)
15. Zaremba, W., Sutskever, I.: Learning to execute. arXiv preprint arXiv:1410.4615 (2014)
16. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
17. Poole, B., Sohl-Dickstein, J., Ganguli, S.: Analyzing noise in autoencoders and deep networks. arXiv preprint arXiv:1406.1831 (2014)
18. Xie, Z., Wang, S.I., Li, J., Lévy, D., Nie, A., Jurafsky, D., Ng, A.Y.: Data noising as smoothing in neural network language models. arXiv preprint arXiv:1703.02573 (2017)
19. Rink, B., Harabagiu, S.: UTD: classifying semantic relations by combining lexical and semantic resources. In: Proceedings of the 5th International Workshop on Semantic Evaluation, pp. 256–259. Association for Computational Linguistics (2010)
20. Plank, B., Moschitti, A.: 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, vol. 1, pp. 1498–1507 (2013)
21. Bunescu, R.C., Mooney, R.J.: A shortest path dependency kernel for relation extraction. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, pp. 724–731. Association for Computational Linguistics (2005)
22. Socher, R., Huval, B., Manning, C.D., Ng, A.Y.: 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, pp. 1201–1211. Association for Computational Linguistics (2012)

23. Miwa, M., Bansal, M.: End-to-end relation extraction using LSTMs on sequences and tree structures. arXiv preprint arXiv:1601.00770 (2016)
24. Yu, M., Gormley, M., Dredze, M.: Factor-based compositional embedding models. In: NIPS Workshop on Learning Semantics, pp. 95–101 (2014)