Relation Classification: CNN or RNN?

Dongxu Zhang[1,3] and Dong Wang*[1,2]

[1] CSLT, RIIT, Tsinghua University
[2] Tsinghua National Lab for Information Science and Technology
[3] PRIS, Beijing University of Posts and Telecommunications
Beijing, China
wangdong99@mails.tsinghua.edu.cn

Abstract. Convolutional neural networks (CNN) have delivered competitive performance on relation classification, without tedious feature engineering. A particular shortcoming of CNN, however, is that it is less powerful in modeling long-span relations. This paper presents a model based on recurrent neural networks (RNN) and compares the capabilities of CNN and RNN on the relation classification task. We conducted a thorough comparative study on two databases: one is the popular SemEval-2010 Task 8 dataset, and the other is the KBP37 dataset we designed based on MIML-RE [1], with the goal of learning and testing complex relations. The experimental results strongly indicate that even with a simple RNN structure, the model can deliver much better performance than CNN, particularly for long-span relations.

1 Introduction

This paper focuses on the task of sentence-level relation classification. Given a sentence X which contains a pair of nominals $\langle x, y \rangle$, the goal of the task is to predict relation $r \in R$ between the two nominals x and y, where R is a set of pre-defined relations [4]. Relation classification is a fundamental task in natural language processing (NLP) and plays an important role in information extraction and knowledge graph construction [9, 11].

A multitude of studies have been conducted on deep neural models in relation classification. Most representative progress was made by Zeng et al. [15], who was motivated by the work of Collobert et al. [3] directly and used a CNN model to extract local semantic patterns and applied them to classify relations. The authors reported quite competitive results without utilizing any extra knowledge resource and NLP modules. Following this success, several CNN-based variants have been recently proposed, including multi-window CNN [5], CR-CNN [7] and NS-depLCNN [13].

The success of CNN models in relation classification can be largely attributed to its capability of learning local semantic patterns, thanks to the flexible convolutional structure of CNNs in multi-dimensional feature extraction. However, CNN possesses a clear shortage in modeling distant dependencies among words, due to the limited size of the convolution window. For the task of relation classification, the semantic meaning of a relation is often embedded in a long sequence of contextual words. With CNN models, the long-span relation has to be split into local word fragments (the length equal to the size of the convolution window), and then the entire semantic is learned by merging the patterns learned from these fragments, where the simple max or mean pooling are often used as the merging algorithm. This splitting-and-pooling approach is certainly not ideal: the learning is separated in two stages and the behavior of the learned CNN is largely impacted by the size of the convolution window, i.e., whether the convolution size matches the length of the local patterns. Additionally, the max or mean pooling does not consider the order of the local patterns learned by the convolution layer.

A number of approaches have been proposed to improve CNN models in learning long-span relations. For example, Zeng et al. [15] proposed to use a position feature that specifies the position of each contextual word relative to the two target nominals. This position feature could provide order information of the contextual words, and was reported to offer significant performance improvement. Nguyen and Grishman [5] proposed a CNN structure with multiple convolution windows, which allows learning patterns of multiple levels of granularity. This approach, however, involves much more computation, and selecting the sizes of the convolution windows is not trivial. Another approach is to find the key path of the input sequence so that long-span relations can be learned on just a few key words on the key path. For example, the shortest dependency path approach presented in [13, 9]. A shortcoming of this approach is that it depends on a parser that requires much computation and is error-prone. Moreover, if the dependency path is long, the problem associated with CNNs still exists.

In contrast to CNN-based methods, there are also a few work [14, 11] using RNNs with long-short term memory(LSTM) units. P. Verga et al. [11] show that LSTMs outperforms CNNs in the knowledge base inferring task. But in the task of sentence-level relation classification, CNN-based models and basic RNN-based models have not been compared thoroughly with equal conditions.

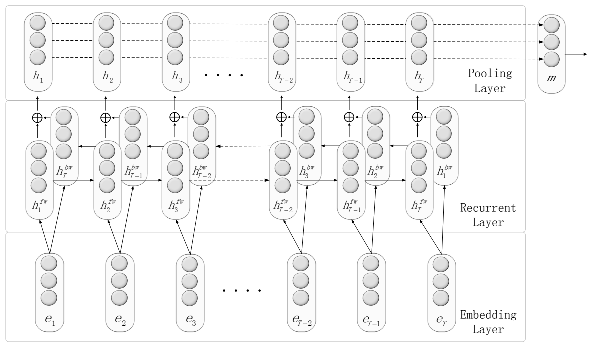

Fig. 1. The structure of the proposed RNN-based model.

In this paper, we analyse the capability of RNN on the relation classification task. Compared with CNN, RNN can memorize a long history and so is good at modeling sequential data [2]. To employ RNN in relation learning, a few modifications are proposed in our study, including a bi-directional architecture and a position indicator feature. The structure of the RNN-based model is shown in Figure 1 and some details will be described in Section 2. The main contributions of this paper are as follows:

– Designed a new dataset based on the MIML-RE's annotated data [1] that involves more complex relations, and verified the advantage of RNN on the new dataset.
– Empirically analyzed the advantage of the RNN-based approach in learning long-span relations.
– Demonstrated that the position indicator feature proposed in this work is more effective than the position feature proposed by Zeng at al. [15].

2 Model

As has been shown in Figure 1, the RNN-based model used in this paper contains three components: (1) a word embedding layer that maps each word in a sentence into a low dimension word vector; (2) a bidirectional recurrent layer that models the word sequence and produces word-level features (representations); (3) a max pooling layer that merges the word-level features of all the time steps into a sentence-level vector, by selecting the maximum value among all the word-level features for each dimension. The sentence-level vector is finally used for relation classification.

2.1 Model Training

The training objective is to let the produced sentence vectors maximize performance on the task of relation classification. To keep consistency between our model and CNN model proposed by Zeng et al. [15], here we use a simple softmax regression as the classifier and objective function is the cross entropy between the predictions and the labels. And we follow the training method proposed by Collobert et al. [3], and utilizes the stochastic gradient descent (SGD) algorithm. Specifically, the back propagation through time (BPTT) [12] is employed to compute the gradients layer by layer, and the fan-in technique proposed by Plaut and Hinton [6] is used to initialize the parameters.

2.2 Position Indicators

In relation learning, it is essential to let the algorithm know the target nominals. In the CNN-based approach, Zeng et al. [15] augment each word vector a position feature vector, i.e., the feature that indicates the distance from each word to the two nominals. This technique has been found highly important to gain high classification accuracy and was followed by a number of studies [5, 7]. For RNN, since the model learns the entire word sequence, the *relative* position information for each word can be obtained automatically in the forward or backward recursive propagation. It is therefore sufficient to annotate the target nominals in the word sequence, without necessity to change the feature vector.

We choose a simple method that uses four position indicators (PI) to specify the starting and ending of the nominals. The following is an example: "<e1> **people** </e1> have been moving back into <e2> **downtown** </e2>". Note that **people** and

downtown are the two nominals with the relation 'Entity-Destination (e1,e2)', and <e1>, </e1>, <e2>, </e2> are the four position indicators which are regarded as single words in model training and test. The position-augmented sentences are then used to train the RNN model as usual. Compared with the position feature approach, the position indictor approach is more straightforward. In Section 3.3, we will show that in most circumstances, the position indictor approach leads to better performance.

3 Experiments

3.1 Database

Table 1. Statistics of KBP37 .

Number of training data	15917
Number of development data	1724
Number of test data	3405
Number of relation types	37
per:alternate_names	org:alternate_names
per:origin	org:subsidiaries
per:spouse	org:top_members/employees
per:title	org:founded
per:employee_of	org:founded_by
per:countries_of_residence	org:country_of_headquarters
per:stateorprovinces_of_residence	org:stateorprovince_of_headquarters
per:cities_of_residence	org:city_of_headquarters
per:country_of_birth	org:members
no_relation	

We use two datasets to evaluate the proposed RNN model. The first one is the dataset provided by SemEval-2010 Task 8. In this dataset, there are regular 9 relation types and an additional 'other' relation. Taking the direction into account (the order of the two nominals in the relation), this dataset involves 19 relation classes in total. The dataset is publicly available.[4]

The second dataset is constructed by the authors based on the MIML-RE annotation dataset provided by Angeli et al. [1]. The original MIML-RE dataset involves the 2010 and 2013 KBP official document collections, plus a July 2013 dump of Wikipedia. There original data collection consists of 33811 annotated sentences. To make the dataset more suitable for relation classification tasks, we made several refinements:

1. First, we add two directions to each relation type, except the type 'no_relation'. For example, 'per:employee_of' is split into two relations 'per:employee_of(e1,e2)' and

[4] `http://docs.google.com/View?docid=dfvxd49s_36c28v9pmw`

'per:employee_of(e2,e1). To avoid incorrect accounting of errors associated with the directional relations[5], we replace 'org:parents(e1,e2)' with 'org:subsidiaries(e2,e1)' and replace 'org:member_of(e1,e2)' with 'org:member(e2,e1)'. After this arrangement, there are 76 relations in total.

2. Second, we compute the frequency of each relation type. Only the relation types with a frequency higher than 100 in both directions are retained. This results in 19 relation types, including the 'no_relation' type. Each relation type (except 'no_relation') corresponds to two relation classes, by considering the relation direction. This leads to 37 relation classes in total. To gain better data balance, 80% of the sentences in the 'no_relation' type are randomly discarded.
3. Finally, the entire dataset is randomly split into three subsets: 70% for training, 10% for development, and 20% for testing. For a more strict test, sentences in the development and test set are removed if the nominal pairs *and* their relation have been both found in one training sentence.

In the rest of the paper, we will call the second dataset KBP37. Some statistics of this dataset and all the relation types are shown in Table 1. Notice that KBP37 is different from SemEval-2010 Task 8 in several aspects: First, there are more relation types in KBP37, and the relations are more 'specific' compared with those in SemEval-2010 task 8. For example, 'Member Collection' in SemEval-2010 task 8 can represent any 'member of' relation, while 'org:member' in KBP specifically refers to the relation between mother and child companies. Second, the nominal pairs in KBP37 are all entity names and so are more sparse than SemEval-2010 task 8 in both the domain-independent and task-specific corpus. Third, there are much more multi-word nominals in KBP37. Finally, the average length of the sentences in KBP37 is much longer than Semeval-2010 task 8, which will be discussed in details in Section 4.2.

It can be seen that the relation classification task on KBP37 is more difficult but also more realistic. We argue that SemEval-2010 Task 8 has been studied for a long time and the performance has been over-tuned. Therefore to evaluate a new algorithm, it would be more informative to test it on a new and more complex dataset. We design KBP37 to meet this request, particularly for learning and test on long-span relations. This dataset can be downloaded publicly[6].

3.2 Experimental Setup

In order to compare with the work by Socher et al. [8] and Zeng et al. [15], we use the same 50-dimensional word vectors proposed by Turian et al. [10] to initialize the embedding layer in the experiments.

Because there is no official development dataset in Semeval-2010 Task 8 dataset, we tune the hyper-parameters by 8-fold cross validation. Once the hyper-parameters are optimized, all the 8000 training data are used to train the model with the best configuration. With Turian's 50-dimensional word vectors, the best dimension of the sentence

[5] This is mainly caused by some relations that have been directional already in KBP. See the KBP manual at `http://surdeanu.info/kbp2013/TAC_2013_KBP_Slot_Descriptions_1.0.pdf`

[6] `https://github.com/zhangdongxu/kbp37`

vector is 800. The number of iterations is set to 20. For fast convergence, we set the learning rate to 0.1 in the first five iterations, and then diminish it to 0.01.

For KBP37, the hyper-parameters are tuned based on the development set. The development data is also used to choose the best model among different iterations. With Turian's 50-dimensional word vectors, the best dimension of the sentence vector is 700. The training process is the same as the one used on the Semeval-2010 Task 8 dataset.

3.3 Results

The performance is evaluated in terms of F1 score defined by SemEval-2010 Task 8 [4]. Note that given a sentence and two target nominals, a prediction is counted as correct only when both the relation type and its direction are correct.

Table 2. F1 results with the proposed RNN model on SemEval-2010 Task 8 dataset, '+' means adding a new modification.

Model	RNN	+ max-pooling	+ position indicators	+ bidirection
F1	31.9	67.5	76.9	79.6

Table 3. Comparison of F1 scores with different neural models on different datasets. The 50-dimensional word vectors provided by Turian et al. [10] are used for pre-training. 'PF' stands for position features and 'PI' stands for position indicators. The numbers in the parentheses show the best dimensions of the hidden layer. Since Zeng et al. [15] did not release the source code, we reproduced their method, denoted by 'Our rep.'.

Model	MV-RNN (Socher,2012)	CNN+PF (Zeng,2014)	CNN+PF (Our rep.)	CNN+PI (Our rep.)	RNN+PF	RNN+PI
Semeval-2010 task8	79.1	78.9	78.3 ($300 \to 300$)	77.4 ($400 \to 400$)	78.8 (400)	**79.6** (800)
KBP37	-	-	51.3 ($500 \to 500$)	55.1 ($500 \to 500$)	54.3 (400)	**58.8** (700)

Table 2 presents the F1 results of the proposed RNN model on the SemEval-2010 Task 8 dataset, with the contribution offered by each modification. It can be seen that the basic RNN, which is single directional and with the output of the last step as the sentence vector, performs poorly. This can be attributed to the lack of the position information of target nominals and the difficulty in RNN training. The max-pooling offers the most significant performance improvement, indicating that local patterns learned from neighbouring words are highly important for relation classification. The position indicators also produce highly significant improvement, which is not surprising as without the position information, the model would be puzzled by which pattern to learn. The contribution of positional information has been demonstrated by Zeng et al. [15], where the positional features lead to nearly 10 percentiles of F1 improvement, which is similar as the gain obtained in our study.

The second experiment compares three representative neural models, especially CNN and RNN. The results are presented in Table 3, where the 50-dimensional word vectors are employed in the experiments. Since Zeng et al. [15] did not release the source code, we tried to reproduce their result and got a very close number although not identical (78.3 vs. 78.9), as shown in the third row of Table 3.

Comparing the results with different models and methods, it can be seen that the RNN model outperforms both the MV-RNN model proposed by Socher et al. [8] and the CNN model proposed by Zeng et al. [15]. This demonstrates that an RNN model is more appropriate for the relation classification task. An interesting observation is that the RNN model performs better than the MV-RNN model which uses syntactic parsing as extra resources. This indicates that relation patterns can be effectively learned by RNNs from raw text, without any explicit linguistic modules and knowledge.

Comparing the results on the two datasets, it can be observed that the F1 scores are much lower on KBP37 than on Semeval-2010 Task 8. This indicates that KBP37 is a more difficult dataset. Moreover, the RNN model beats the CNN by a larger margin on KBP37. This indicates that RNN possesses more advantage in learning more complex relations. More discussion will be given in Section 4.2.

From Table 3, we also find that with RNN, position indicators (PI) is more effective than position features (PF), and this is more evident on the KBP37 task. A possible reason is that the recurrent process can remember the PI vectors but may corrupt the PF information associated with each word. More discussion will be presented in Section 4.1.

4 Discussion

4.1 Impact of Long Context

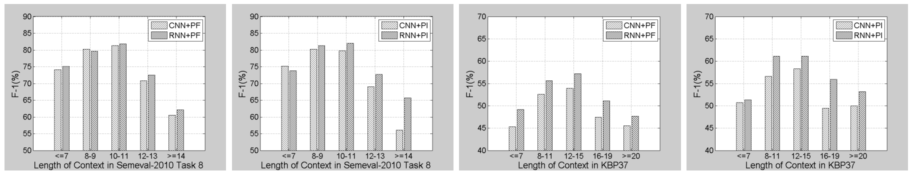

Fig. 2. F1 scores with different lengths of contexts on Semeval-2010 Task 8 and KBP37.

We have argued that a particular advantage of the RNN model compared with the CNN model is that it can learn long-distance patterns more effectively, and so can better deal with long-span relations. To verify this argument, we split each test dataset (Semeval-2010 Task8 and KBP37) into 5 subsets according to the context length of the relations, i.e., the number of words between the two nominals, plus 2 words prior to the first nominal and 2 words after the second nominal, if they exist. The position indicators are not counted.

The F1 results on the 5 subsets are reported in Figure 2. It can be seen that on both test datasets, if the context length is small, the CNN and RNN models perform similar with PI, whereas if the context length is large, the RNN model is clearly superior. This confirms that RNN is more suitable to learn long-distance patterns. If PF is used, this trend is not such clear. This seems to indicate that PF is more suitable for CNN and PI is more suitable for RNN. Nevertheless, PI performs much better than PF in KBP37 even with CNN (51.3 vs. 55.1 from Table 3), which suggests that PI tends to be more robust.

Note that with both the two models, the best F1 results are obtained with a moderate context length, no matter how the position information is used. This is understandable, as too small contexts involve limited semantic information, while too large contexts lead to difficulties in pattern learning.

4.2 Proportion of Long Context

Table 4. The distribution of context lengths with two datasets.

Dataset	Context Length			Proportion of Long Context ($\geq$ 11)
	$\leq$10	11 - 15	$\geq$ 16	
SemEval-2010 task-8	6658	3725	334	0.379
KBP37	4719	7512	8815	0.776

Figure 2 shows that the RNN model significantly outperforms the CNN model on almost all the configurations. This is a little different from the results presented in Table 3, where the discrepancy between the two models in SemEval-2010 dataset is not so remarkable (78.8 vs. 78.3 with PF and 77.4 vs. 79.6 with PI). This can be attributed to the small proportion of long contexts in test data of SemEval-2010 dataset.

To make it clear, the distribution of the context length is calculated on the two datasets. The statistics are shown in Table 4. It can be observed that long contexts exist in both datasets, but the proportion of long contexts in the SemEval-2010 task 8 dataset is much smaller than those in the KBP dataset. This suggests that the strength of a method can not be fully demonstrated on the SemEval-2010 task 8 dataset. This is the reason we release KBP37 and argue that it is a major contribution of this paper.

4.3 Semantic Accumulation

Another interesting analysis is to show how the 'semantic meaning' of a sentence is formed. First notice that with both the CNN and the RNN models, the sentence-level vectors are produced from local features (word-level for CNN and segment-level for RNN) by dimension-wise max-pooling.

To measure the contribution of a particular word or segment to the sentence-level semantic meaning, for each sentence, we count the number of dimensions that the local

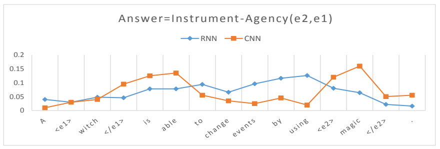

Fig. 3. Semantic distribution on words in the sentence "A <e1> witch </e1> is able to change events by using <e2> magic </e2> ."

feature at each word step contributes to the output of the max-pooling. This number is divided by the number of total dimensions of the feature vector, resulting in a 'semantic distribution' over the word sequence. Figure 3 shows an example of the semantic distribution with results from both the CNN and RNN models.

From Figure 3, the correct relation is 'Instrument-Agency', but CNN gives wrong answer 'Other'. It can be seen that CNN matches two patterns 'is able to' and 'magic', while the RNN matches the entire sequence between the two nominals **witch** and **magic**, with the peak at 'by using'. Clearly, the pattern that the RNN model matches is more reasonable than that matched by the CNN model. We highlight that RNN is a temporal model which accumulates the semantic meaning word by word, so the peak at 'by using' is actually the contribution of all the words after 'witch'. In contrast, CNN model learns only local patterns, therefore it splits the semantic meaning into two separate word segments.

An interesting observation is that the RNN-based semantic distribution tends to be smoother than the one produced by the CNN model. In fact, we calculate the average variance on the semantic contribution of neighbouring words with all the sentences in the SemEval-2010 task 8 dataset, and find out that the variance with the RNN model is 0.0017, while this number is 0.0025 with the CNN model. The smoother semantic distribution is certainly due to the temporal nature of the RNN model.

5 Conclusion

In this paper, we compare CNN-based and RNN-based approaches for relation classification. Since the RNN model can deal with long-distance patterns, it is particular suitable for learning long-span relations. Several important modifications were proposed to improve the basic RNN model, including a max-pooling feature aggregation, a position indicator approach to specify target nominals, and a bi-directional architecture to learn both forward and backward contexts. Experimental results on two datasets demonstrated that the RNN-based approach can achieve better results than CNN-based approach, and for sentences with long-span relations, the RNN model exhibits clear advantage. Last but not least, we released a new dataset KBP37 that contains more complex patterns and long-span relations, therefore more suitable for evaluating models and methods for relation classification.

Acknowledgments

This research was supported by the National Science Foundation of China (NSFC) under the project No. 61371136, and the MESTDC PhD Foundation Project No. 20130002 120011. It was also supported by Huilan Ltd.

References

1. Angeli, G., Tibshirani, J., Wu, J.Y., Manning, C.D.: Combining distant and partial supervision for relation extraction. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (October 2014)
2. Boden, M.: A guide to recurrent neural networks and backpropagation. In: the Dallas project, SICS Technical Report T2002:03 (2002)
3. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research 12, 2493–2537 (2011)
4. 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 (2009)
5. Nguyen, T.H., Grishman, R.: Relation extraction: Perspective from convolutional neural networks. In: Proceedings of the 2013 conference of NAACL Workshop on Vector Space Modeling for NLP (2015)
6. Plaut, D.C., Hinton, G.E.: Learning sets of filters using back-propagation. Computer Speech & Language 2(1), 35–61 (1987)
7. dos Santos, C.N., 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. pp. 626–634 (2015)
8. Socher, R., Huval, B., Manning, C.D., Ng, A.Y.: Semantic compositionality through recursive matrix-vector spaces. In: Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (2012)
9. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (2015)
10. Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th annual meeting of the association for computational linguistics. pp. 384–394 (2010)
11. Verga, P., Belanger, D., Strubell, E., Roth, B., McCallum, A.: Multilingual relation extraction using compositional universal schema. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics (2016)
12. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE 78(10), 1550–1560 (1990)
13. 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 (2015)
14. Yan, X., Mou, L., Li, G., Chen, Y., Peng, H., Jin, Z.: Classifying relations via long short term memory networks along shortest dependency path. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (2015)
15. Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J.: Relation classification via convolutional deep neural network. In: Proceedings of COLING. pp. 2335–2344 (2014)