CSLT TECHNICAL REPORT-20150024 [Friday 16th October, 2015]

Relation Classification via Recurrent Neural Network

Dong Xu Zhang[1], Dong Wang[1*] and Rong Liu[1]

[*]Correspondence: wang-dong99@mails.tsinghua.edu.cn
[1]Center for Speech and Language Technology, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China
Full list of author information is available at the end of the article

Abstract

Deep learning has gained much success in sentence-level relation classification. For example, convolutional neural networks (CNN) have delivered state-of-the-art performance without much effort on feature engineering as the conventional pattern-based methods. A key issue that has not been well addressed by the existing research is the lack of capability to learn temporal features, especially long-distance dependency between nominal pairs. In this paper, we propose a novel framework based on recurrent neural networks (RNN) to tackle the problem, and present several modifications to enhance the model, including a max-pooling approach and a bi-directional architecture. Our experiment on the SemEval-2010 Task-8 dataset shows that the RNN model can deliver state-of-the-art performance on relation classification, and it is particularly capable of learning long-distance relation patterns. This makes it suitable for real-world applications where complicated expressions are often involved.

Keywords: relation learning; recurrent neural network

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 [1].

Conventional relation classification methods are mostly based on pattern matching, and an obvious disadvantage is that high-level features such as tags of part of speech (POS) and name entities are often involved. These high-level features require extra NLP modules that not only increase computational cost, but introduce additional errors. Also, manually designing patterns is always time-consuming with low coverage.

Recently, deep learning has made significant progress in natural language processing. [2] proposed a general framework which derives task-oriented features by learning from raw text data using convolutional neural networks (CNN). The idea of 'learning from scratch' is fundamentally different from the conventional methods which require careful and tedious feature engineering. [2] evaluated the learning-based approach on several NLP tasks including POS tagging, NER and semantic role labelling. Without any human-designed features, they obtained close to or even better performance than the state-of-the-art systems that involve complicated feature engineering.

A multitude of researches have been proposed to apply the deep learning methods and neural models to relation classification. Most representative progress was made

by [3], who proposed a CNN-based approach that can deliver quite competitive results without any extra knowledge resource and NLP modules. And there are other valuable models such as MV-RNN[4], CNN[5], FCM[6].

Despite the success obtained so far, most of the current deep learning approaches to relation learning and classification are weak in modeling temporal patterns. Note that the semantic meaning of a relation is formed in the context of the two target nominals, including the word sequence between them and a window of preceding and following words. Additionally, the relation is in fact 'directional', which means the order of the context words does matter. Therefore, relation learning is essentially a task of temporal sequence learning, and so should be modelled by a temporal model. Most of the current deep learning models mentioned above are static models, and are potentially weak especially when learning long-distance relation patterns. For example, the CNN model can learn only local patterns, and so is hard to deal with patterns that is outside of the window of the convolutional filter.

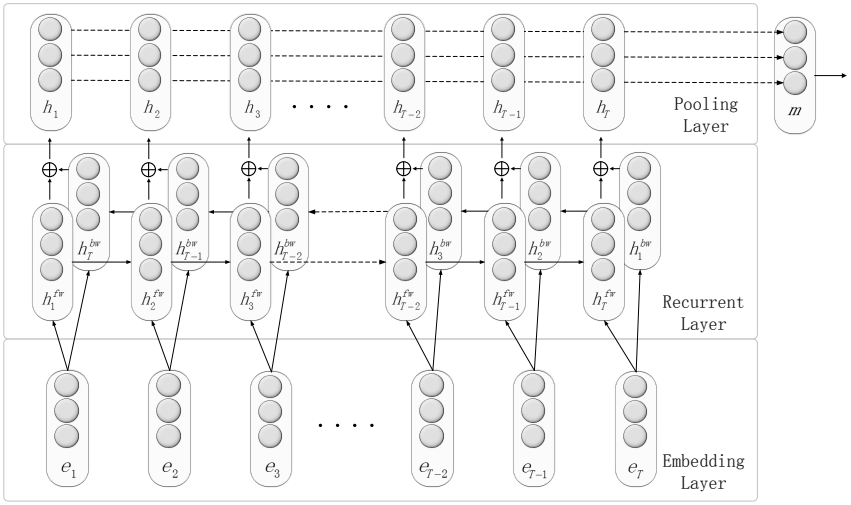

Figure 1 The framework of the proposed model.

In this paper, we propose a novel framework based on recurrent neural networks (RNN) to tackle the problem of long-distance pattern learning. Compared to other models such as CNN, RRN is a temporal model and is particularly good at modeling sequential data [7]. The main framework is shown in Figure 1, which will be described in details in Section 3.

The main contributions of this paper are as follows:

- Proposed an RNN-based framework to model long-distance relation patterns.
- Verified the proposed approach on the SemEval-2010 task-8 dataset and obtained state-of-the-art performance.
- Analyzed empirically the capability of the RNN-based approach in modeling long-distance patterns.

2 Related Work

As mentioned, the conventional approaches to relation classification are based on pattern matching, and can be categorized into feature-based methods [8, 9] and kernel-based methods [10, 11]. The former category relies on human-designed patterns and so require expert experience and time consuming, and the latter category suffers from data sparsity. Additionally, these methods rely on extra NLP tools to derive linguistic features.

To alleviate the difficulties in pattern design and also the lack of annotated data, distant supervision has drawn a lot of attention since 2009 [12, 13, 14, 15]. This technique combines resources of text data and knowledge graph, and uses the relations in the knowledge graph to discover patterns automatically from text data. However they still depend on NLP tools.

Our work follows the line of automatic feature learning by neural models, which is largely fostered by [2]. A closely related work was proposed by [3], which employed CNN to learn patterns of realtions from raw text data and so is a pure feature learning approach. A potential problem of CNN is that this model can learn only local patterns, and so is not suitable for learning long-distance patterns in relation learning. Particularly, simply increasing the window size of the convolutional filters does not work: that will lose the strength of CNNs in modeling local or short-distance patterns. To tackle this problem, [5] proposed a CNN model with multiple window sizes for filters, which allows learning patterns of different lengths. Although this method is promising, it involves much more computation, and tuning the window sizes is not trivial. The RNN-based approach solves the difficulty of CNN models in learning long-distance and variable-distance patterns in an elegant way.

Another related work is MV-RNN model proposed by [4]. The difference is that we based on different RNNs: the MV-RNN model is based on recursive NN while our work is based on recurrent NN, a temporal model. Additionally, MV-RNN relies on syntactic parsing, and our model uses only word vectors and so is more efficient especially in predicting process.

Finally, our work related to the FCM framework proposed recently [6]. In principle, FCM decomposes sentences into substructures and factorizes semantic meaning into contributions from multiple annotations (e.g., POS, NER, dependency parse). It can be regarded as a general form of the MV-RNN and CNN models where the recursive hierarchy or max-pooling are replaced by a general composition function. Nevertheless, FCM is still a static model and shares the same disadvantage of CNN in modeling temporal data.

The advantage of the RNN model in learning sequential data is well-known and has been utilized in language modeling [16] and sequential labeling [17]. Compared to these studies, a significant difference of our model is that there are no predicting targets at each time step, and the supervision (relation label) is only available at the end of a sequence. This is similar to the semantic embedding model proposed by [18], though we have made several important modifications, as will be presented in the next section.

3 Model

As has been shown in Figure 1, the model proposed 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 word-level features from each time step (each word) into a sentence-level feature vector, by selecting the maximum value among all the word-level features for each dimension. The sentence-level feature vector is finally used for relation classification. These components will be presented in detail in this section.

3.1 Word embedding

The word embedding layer is the first component of the proposed model, which projects discrete word symbols to low-dimensional dense word vectors, so that the words can be modeled and processed by the following layers. Let $x_t \in \{0,1\}^{|V|}$ denote the one-hot representation of the t-th word v_t, where $|V|$ denotes the size of the vocabulary V. The embedding layer transfers x_t to word vectors $e_t \in R^D$ as follows:

$$e_t = W_{em} x_t \tag{1}$$

where $W_{em} \in R^{|D| \times |V|}$ is the projection matrix. Since x_t is one-hot, W_{em} in fact stores representations of all the words in V. Word embedding has been widely studied in the context of semantic learning. In this work, we first train word vectors using the word2vec tool[1] with a large amount of data that are in general domains, and then use these vectors to initialize (pre-train) the word embedding layer of our model. By this way, knowledge of general domains can be used. It has been shown that this pre-training improves model training, e.g., [3, 6].

3.2 Bi-directional network

The second component of our model is the recurrent layer, the key part for modeling sequential data and long-distance patterns. We start from a simple one-directional forward RNN. Given a sentence $X = (x_1, x_2, ..., x_T)$, the words are projected into a sequence of word vectors, denoted by $(e_1, e_2, ..., e_T)$ where T is the number of words. These word vectors are put to the recurrent layer step by step. For each step t, the network accepts the word vector e_t and the output at the previous step h_{t-1}^{fw} as the input, and produces the current output h_t^{fw} by a linear transform followed by a non-linear activation function, given by:

$$h_t^{fw} = \tanh(W_{fw} e_t + U_{fw} h_{t-1}^{fw} + b_{fw}) \tag{2}$$

where $h_t^{fw} \in R^M$ is the output of the RNN at the t-th step, which can be regarded as local segment-level features produced by the word segment $(x_1, ..., x_t)$. Note that M is the dimension of the feature vector, and $W_{fw} \in R^{M \times D}$, $U_{fw} \in R^{M \times M}$, $b_{fw} \in R^{M \times 1}$ are the model parameters. We have used the hyperbolic function $\tanh(\cdot)$ as the non-linear transform, which can help back propagate the error more easily due to its symmetry [19].

[1] http://code.google.com/p/word2vec/

A potential problem of the one-directional forward RNN is that the information of future words are not fully utilized when predicting the semantic meaning in the middle of a sentence. A possible solution is to use a bi-directional architecture that makes predictions based on both the past and future words, as has been seen in Figure. 1. This architecture has been demonstrated to work well in sequential labeling, e.g., [17]. With the bi-directional RNN architecture, the prediction at step t is obtained by simply adding the output of the forward RNN and the backward RNN, formulated as follows:

$$h_t = h_t^{fw} + h_t^{bw} \tag{3}$$

where $h_t^{bw} \in R^M$ is the output of the backward RNN, which possesses the same dimension as h_t^{fw} defined by:

$$h_t^{bw} = \tanh(W_{bw} e_t + U_{bw} h_{t+1}^{bw} + b_{bw}) \tag{4}$$

where $W_{bw} \in R^{M \times D}$, $U_{bw} \in R^{M \times M}$, $b_{bw} \in R^{M \times 1}$ are the parameters of the backward RNN. Note that the forward and backward RNNs are trained simultaneously, and so the addition is possible even without any parameter sharing between the two RNN structures.

3.3 Max-pooling

Sentence-level relation classification requires a single sentence-level feature vector to represent the entire sentence. In the CNN-based models, a pooling approach is often used [3]. With the RNN structure, since the semantic meaning of a sentence is learned word by word, the segment-level feature vector produced at the end of the sentence actually represents the entire sentence. This accumulation approach has been used in [18] for sentence-level semantic embedding.

In practice, we found that the accumulation approach is not very suitable for relation learning because there are many long-distance patterns in the training data. Accumulation by recurrent connections tends to forget long-term information quickly, and the supervision at the end of the sentence is hard to be propagated to early steps in model training, due to the annoying problem of gradient vanishing [20].

We therefore resort to the max-pooling approach as in CNN models. The argument is that the segment-level features, although not very strong in representing the entire sentence, can represent local patterns well. The semantic meaning of a sentence can be achieved by merging representations of the local patterns. The max-pooling is formulated as follows:

$$h_i = \max_t \{(h_t)_i\}, \quad \forall i = 1, ..., M \tag{5}$$

where h is the sentence feature vector and i indexes feature dimensions.

Note that we have chosen max-pooling rather than mean-pooling. The hypothesis is that only several key words (trigger) and the associated patterns are important

for relation classification, and so max-pooling is more appropriate to promote the most informative patterns.

3.4 Model training

Training the model in Figure 1 involves optimizing the parameters $\theta = \{W_{in}, W_{fw}, U_{fw}, b_{fw}, W_{bw}, U_{bw}, b_{bw}\}$. The training objective is that, for a given sentence, the output feature vector h achieves the best performance on the task of relation classification. Here we use a simple logistic regression model as the classfier. Formally, this model predicts the posterior probability that an input sentence X involves a relationship r as follows:

$$P(r|X, \theta, W_o, b_o) = \sigma(W_o h(X) + b_o) \tag{6}$$

where $\sigma(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$ is the softmax function, and θ encodes the parameters of the RNN model.

Based on the logistic regression classifier, a natural objective function is the cross entropy between the predictions and the labels, given by:

$$\mathcal{L}(\theta, W_o, b_o) = \sum_{n \in N} -\log p(r^{(n)}|X^{(n)}, \theta, W_o, b_o) \tag{7}$$

where n is the index of sentences in the training data, and $X^{(n)}$ and $r^{(n)}$ denote the n-th sentence and its relation label, respectively.

To train such a model, we follow the training method proposed by [2], and utilizes the stochastic gradient descent (SGD) algorithm. Specifically, the back propagation through time (BPTT) [21] is employed to compute the gradients layer by layer, and the fan-in technique proposed by [22] is used to initialize the parameters. It was found that this initialization can locate the model parameters around the linear region of the activation function, which helps propagating the gradients back to early steps easier. Moreover, it also balances the learning speed for parameters in different layers [23].

As has been discussed, pre-training the word embedding layer with word vectors trained from extra large amount corpus improves the performance. This approach has been employed in our experiments.

3.5 Position indicators

In relation learning, it is essential to let the algorithm know the target nominals. In the CNN-based approach, [3] appended a position feature vector to each word vector, i.e., the distance from the word to the two nominals. This has been found highly important to gain high classification accuracy. For RNN, since the model learns the entire word sequence, the *relative* positional 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 input vectors.

We choose a simple method that uses four position indicators 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 the training and testing process. The position-embedded sentences are then used as the input to train the RNN model. Compared to the position feature approach in the CNN model, the position indictor method is more straightforward.

4 Experiments

4.1 Database and experimental setup

We use the dataset and evaluation framework provided by SemEval-2010 Task 8. There are 9 *directional* relations and an additional 'other' relation, resulting in 19 relation classes in total. Given a sentence and two target nominals, a prediction is counted as correct only when both the relation and its direction are correct. The performance is evaluated in terms of the F1 score defined by SemEval-2010 Task 8 [1]. Both the data and the evaluation tool are publicly available.[2]

In order to compare with the work by [4] and [3], we use the same word vectors proposed by [24] (50-dimensional) to initialize the embedding layer in the main experiments. Additionally, to compare with the work by [5], additional experiments are also conducted with the word vectors pre-trained by [25] which are 300-dimensional.

Because there is not an official development dataset, we tune the hyperparameters by 8-fold cross validation. Once the hypeparameters are optimized, all the training data are used to train the model with the best configuration. With Turian's 50-dimensional word vectors, the best dimension of the feature vector is $M = 500$, and with Mkolov's 300-dimensional word vectors, the best feature dimension is $M = 700$. The best learning rate is 0.01 in both the two conditions.

4.2 Results

Model	F1
RNN	31.9
+ max-pooling	67.5
+ position indicators	76.9
+ bidirection	80.0

Table 1 F1 results with the proposed RNN model, with contribution of each modification.

Model	Features	F1
MV-RNN [4]	syntactic parse	79.1
CNN [3]	PF	78.9
RNN (proposed)	PI	**80.0**

Table 2 Comparison of F1 scores with different neural models. The 50-dimensional word vectors provided by [24] are used for pre-training. PF stands for position features and PI stands for position indicators.

Table 1 presents the F1 results of our RNN model, with the contribution offered by each modification. It can be seen that the basic RNN, which is signal directional

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

Model	Features Added	F1
SVM	POS, prefixes, morphological, WordNet, dependency parse,	82.2
[1]	Levin classed, ProBank, FrameNet, NomLex-Plus,	
	Google n-gram, paraphrases, TextRunner	
MV-RNN	WV [26](dim=50), syntactic parse	79.1
[4]	+POS, NER, WordNet	82.4
CNN	WV [24](dim=50)	69.7
[3]	+PF	78.9
	+wordnet	82.7
CNN	WV [25](dim=300)	
[5]	multiple window size, optimized PF	**82.8**
FCM	WV (dim=200)	80.6
[6]	+ dependency parse, NER	83.0
RNN	WV [24](dim=50), PI	80.0
(proposed)	WV [25](dim=300), PI	**82.5**

Table 3 Comparison of F1 results with different models. WV stands for word vectors. PF stands for position features, and PI stands for position indicators.

and with the output of the last step as the sentence-level features, performs very poor. 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 the model would be puzzled which pattern to learn without the positional information. The contribution of positional information has been demonstrated by [3], where the positional features lead to nearly 10 percentiles of F1 improvement, which is similar as the gain obtained in our model.

The second experiment compares various neural models by using the 50-dimensional word vectors. The results are presented in Table 2. It can be seen that the RNN model outperforms both the MV-RNN model proposed by [4] and the CNN model proposed by [3]. The most interesting observation is that the RNN model performs better than the MV-RNN model which uses syntactic parse as extra resources. This indicates that relation patterns can be effectively learned by RNNs from raw text, without any explicit linguistic knowledge.

Finally, we compare the RNN model with several representative models that achieve state-of-the-art results in relation classification (see Table 3). The first model is based on SVMs and was proposed by [1]. This model can represent the state-of-the-art pattern-based system. All the other models are based on neural networks, which are MV-RNN [4], CNN [3], CNN with multiple window sizes [5], and FCM [6]. Note that different authors use different features and extra knowledge, which makes it difficult to compare the results directly. Nevertheless, some interesting observations can be obtained. Firstly, the best performance obtained so far is 83.0, which was achieved by the FCM model using 300-dimensional word vectors plus dependency parse and name tags as extra knowledge. Among the learning-from-scratch models, i.e., no extra knowledge and NLP processing involved, the best performance (82.8) is achieved by the CNN model with multiple window sizes. Our RNN model with the same word vectors achieves a very similar result (82.5), and the network structure is simpler. In the next section, we will show that the RNN model possesses more potential in real application with complex long-distance relations.

5 Discussion

5.1 Impact of long context

We have argued that a particular advantage of the RNN model compared to the CNN model is that it can deal with long-distance patterns more effectively. To verify this argument, we split the test dataset of SemEval-2010 task-8 into 7 subsets according to length of the context. Here the context is defined as words between the two nominals plus 3 words prior to the first nominal and 3 words after the second nominal, if they exist. To make the analysis more accurate, short clauses between two commas have removed from the context. Clearly, long contexts lead to long-distance patterns.

In order to compare performance of the RNN and CNN models, we tried to reproduce the CNN-based method proposed by [3]. A little difference is that position indicators are used to specify the target nominals in the reproduction. This modification ensures that the two models learn the same input sequence with the same representation. The F1 results on the 7 subsets are reported in Figure 2. It can be seen that if the context length is small, the CNN and RNN models perform similar, 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. Note that with both the two models, the best F1 results are obtained with a moderate length of contexts. This is understandable as too small context involves limited semantic information, while too large context leads to difficulties in pattern learning.

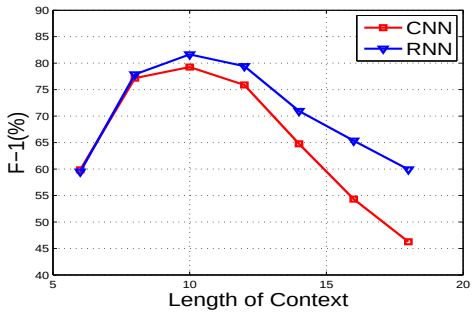

Figure 2 F1 scores with different length of contexts.

5.2 Proportion of long context

Figure 2 shows that the RNN model significantly outperforms the CNN model, which is a little different from the results presented in Table 2, where the discrepancy between the two models is not so remarkable (78.9 vs. 80.0). This can be attributed to the small proportion of long contexts in test data.

To verify this conjecture, the distribution of the context lengths is calculated on the test dataset (SemEval-2010 task8). For comparison, another two datasets are also presented: the New York Time corpus with the entities and relations selected from a subset of Freebased recommended by [27]; the KBP dataset with the modification proposed by [28].

The statistics are shown in Table 4. It can be observed that long contexts exist in all the three datasets. Particularly, the proportion of long contexts in the

Dataset	Context Length			Proportion of Long Context (≥ 11)
	≤ 10	11 - 15	≥ 16	
SemEval-2010 task-8 [1]	6658	3725	334	0.379
NYT+Freebase [27]	22057	19369	3889	0.513
KBP [28]	6618	11647	15546	0.804

Table 4 The distribution of context lengths with three datasets.

SemEval2010 dataset is rather small compared to the other two datasets. This suggests that the strength of the RNN model was not fully demonstrated by our experiments. With a more realistic dataset such as KBP, more advantages are expected to obtain with the RNN model.

5.3 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 features are produced from local features (word-level for CNN and segment-level for RNN) by dimension-wise max-pooling.

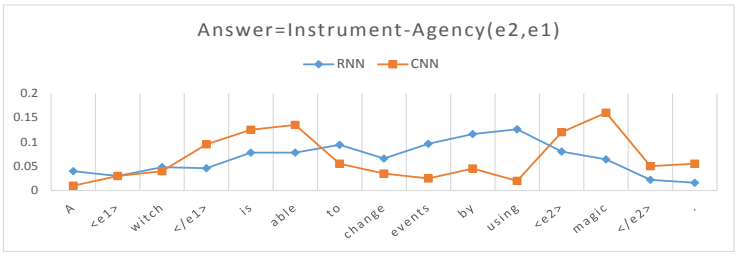

Figure 3 Semantic distribution on words in the sentence "A <e1> witch </e1> is able to change events by using <e2> magic </e2> ."

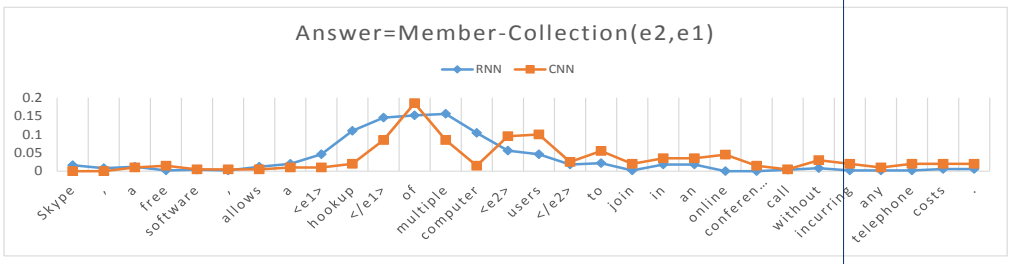

Figure 4 Semantic distribution on words in the sentence "Skype, a free software, allows a <e1> hookup </e1> of multiple computer <e2> users </e2> to join in an online conference call without incurring any telephone costs."

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 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 contribution' over the word sequence. Figure 3 and Figure 4 show

two examples of semantic contributions. In each figure, the results with both the CNN and RNN models are presented.

For the sentence in 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 meanings 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.

Similar observation is obtained with second example shown in Figure 4. Again, the RNN model accumulates the semantic meaning of the sentence word by word, while the CNN model has to learn two local patterns and merge them together.

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 calculated the average variance on the semantic contribution of neighbouring words with all the sentences in the SemEval-2010 task-8 dataset, and found 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.

6 Conclusion

In this paper, we proposed a novel RNN-based approach for relation classification. Compared to other deep learning models such as MV-RNN and CNN, the RNN model can deal with long-distance patterns and so is particular suitable for learning relations within a long context. Several important modifications were proposed to improve the basic model, including a max-pooling feature aggregation, a position indicator approach to specify target nominals, and a bi-directional architecture to learn both the forward and backward contexts. Experimental results demonstrated that the RNN-based approach can achieve very competitive results, and for sentences with long-distance relations, the RNN model exhibits clear advantages.

Acknowledgement

This research was supported by the National Science Foundation of China (NSFC) under the project No. 61371136, and the MESTDC PhD Foundation Project No. 20130002120011. It was also supported by Sinovoice and Huilan Ltd.

Author details
[1]Center for Speech and Language Technology, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China. [2]Center for Speech and Language Technologies, Division of Technical Innovation and Development, Tsinghua National Laboratory for Information Science and Technology, ROOM 1-303, BLDG FIT, 100084 Beijing, China. [3]Department of Computer Science and Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China.

References

1. Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz, "Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," in *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*. Association for Computational Linguistics, 2009, pp. 94–99.
2. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.
3. Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao, "Relation classification via convolutional deep neural network," in *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, 2014, pp. 2335–2344.
4. Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2012.
5. Thien Huu Nguyen and Ralph Grishman, "Relation extraction: Perspective from convolutional neural networks," in *In Proceedings of NAACL Workshop on Vector Space Modeling for NLP*, 2015.
6. Mo Yu, Matthew R Gormley, and Mark Dredze, "Factor-based compositional embedding models," in *NIPS 2014 Workshop*, 2014.
7. Mikael Boden, "A guide to recurrent neural networks and backpropagation," in *In the Dallas project, SICS Technical Report T2002:03*, 2002.
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*. Association for Computational Linguistics, 2004, p. 22.
9. Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum, "Combining linguistic and statistical analysis to extract relations from web documents," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 712–717.
10. Razvan C Bunescu and Raymond J Mooney, "A shortest path dependency kernel for relation extraction," in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2005, pp. 724–731.
11. Raymond J Mooney and Razvan C Bunescu, "Subsequence kernels for relation extraction," in *Advances in neural information processing systems*, 2005, pp. 171–178.
12. 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*. Association for Computational Linguistics, 2009, pp. 1003–1011.
13. Sebastian Riedel, Limin Yao, and Andrew McCallum, "Modeling relations and their mentions without labeled text," in *Machine Learning and Knowledge Discovery in Databases*, pp. 148–163. Springer, 2010.
14. 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*. Association for Computational Linguistics, 2011, pp. 541–550.
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*. Association for Computational Linguistics, 2012, pp. 455–465.
16. Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model.," in *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, 2010, pp. 1045–1048.
17. Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, pp. 2673–2681, 1997.
18. Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward, "Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval," *arXiv preprint arXiv:1502.06922*, 2015.
19. Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.
20. Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.
21. Paul J Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
22. David C Plaut and Geoffrey E Hinton, "Learning sets of filters using back-propagation," *Computer Speech & Language*, vol. 2, no. 1, pp. 35–61, 1987.
23. Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.
24. Joseph Turian, Lev Ratinov, and Yoshua Bengio, "Word representations: a simple and general method for semi-supervised learning," in *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, 2010, pp. 384–394.

25. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.
26. R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *International Conference on Machine Learning, ICML*, 2008.
27. Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin, "Relation extraction with matrix factorization and universal schemas," in *Proceedings of NAACL-HLT*, 2013, pp. 74–84.
28. Gabor Angeli, Julie Tibshirani, Jean Y. Wu, and Christopher D. Manning, "Combining distant and partial supervision for relation extraction," in *EMNLP*, October 2014.