Relation Classification via Recurrent Neural Network

Dongxu Zhang[1,3], **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
wangdong99@mails.tsinghua.edu.cn

Abstract

Deep learning has gained much success in sentence-level relation classification. For example, convolutional neural networks (CNN) have delivered competitive performance without much effort on feature engineering as the conventional pattern-based methods. Thus a lot of works have been produced based on CNN structures. However, a key issue that has not been well addressed by the CNN-based method is the lack of capability to learn temporal features, especially long-distance dependency between nominal pairs. In this paper, we propose a simple framework based on recurrent neural networks (RNN) and compare it with CNN-based model. To show the limitation of popular used SemEval-2010 Task 8 dataset, we introduce another dataset refined from MIML-RE(Angeli et al., 2014). Experiments on two different datasets strongly indicates that the RNN-based model can deliver better 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.

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 (Hendrickx et al., 2009).

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), name entities and dependency path 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. Collobert et al. (2011) 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. Collobert et al. (2011) 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 Zeng et al. (2014), who proposed a CNN-based approach that can deliver quite competitive results without any extra knowledge resource and NLP modules. Following the success of CNN, there are some valuable models such as multi-window CNN(Nguyen and Grishman, 2015), CR-CNN(dos Santos et al., 2015) and NS-depLCNN(Xu et al., 2015a) been proposed recently, which are all based on CNN structure. Though some models also based on other structures like MV-RNN(Socher et al., 2012), FCM(Yu et al., 2014) and SDP-LSTM(Xu et al., 2015b), CNN occupies a leading position.

Despite the success obtained so far, most of the current CNN-based learning approaches to relation learning and classification are weak in mod-

eling temporal patterns. Though SDP-LSTM algorithm utilize recurrent structure on dependency parsing, no further analysis has been shown to compare RNN with CNN models. 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. CNN models 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. Dependency path can alleviate this problem by removing noise compared with natural sequence input (for example NS-depLCNN(Xu et al., 2015a)), but the computation cost and adding error caused by dependency parser is inevitable. And the same limitation still exists when the dependency path is long.

In this paper, we propose a simple framework based on recurrent neural networks (RNN) to tackle the problem of long-distance pattern learning. Compared to CNN models, RNN is a temporal model and is particularly good at modeling sequential data (Boden, 2002). 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 advantages of recurrent structure not only on the most used SemEval-2010 task 8 dataset, but also on a new dataset refined from MIML-RE's annotated data(Angeli et al., 2014), and obtained distinct gain compared with CNN-based model.
- Shows that position feature(PF) proposed by Zeng at al.(2014) is less universal than position indicator(PI).
- 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 (Kambhatla, 2004; Suchanek et al., 2006) and kernel-based methods (Bunescu and Mooney, 2005; Mooney and Bunescu, 2005). 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 (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Angeli et al., 2014). 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.

Our work follows the line of automatic feature learning by neural models, which is largely fostered by Collobert et al. (2011). A closely related work was proposed by Zeng et al. (2014), which employed CNN to learn patterns of relations 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, Nguyen and Grishman (2015) 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 could solve the difficulty of CNN models in learning long-distance and variable-distance patterns in an elegant way.

In order to tackle with long-distance dependency patterns, some works are proposed based on dependency trees which can eliminate irrelevant words in the sentence. One early work is MV-RNN model proposed by Socher et al. (2012). 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

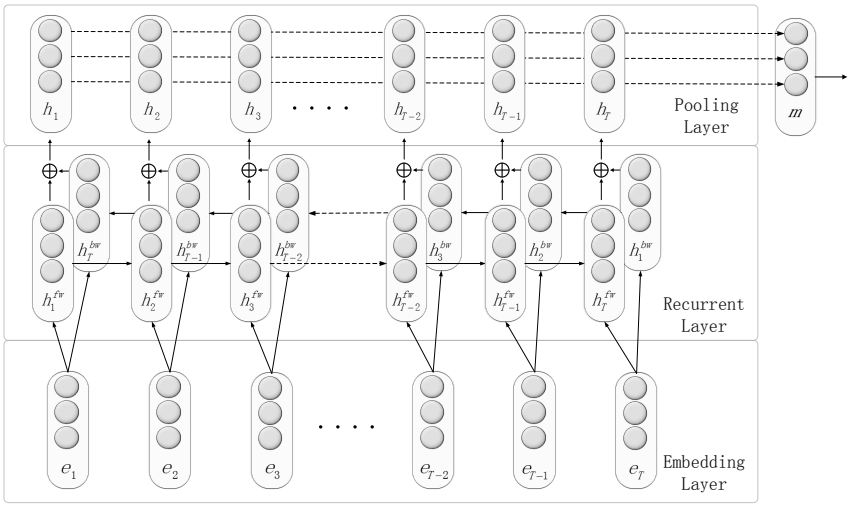

Figure 1: The framework of the proposed model.

model. Recently, Kun Xu at al. (2015a) exploits the dependency path to learn the assignments of subjects and objects using a straightforward negative sampling method, which adopts the shortest dependency path from the object to the subject as a negative sample. More recently, Yan Xu at al. (2015b) proposed a model based on LSTM recurrent neural network, which is most similar with our model. However, these works rely on syntactic parsing, which makes the process more complicated. When sentence becomes longer and the syntax becomes more complex, more error from dependency tree will appear thus influence the final performance.

In addition, our work is related to the FCM framework (Yu et al., 2014). 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. dos Santos et al. (2015) also use the convolutional network. And they propose a ranking-based cost function and elaborately diminish the impact of the Other class.

The advantage of the RNN model in learning sequential data is well-known and has been utilized in language modeling (Mikolov et al., 2010) and sequential labeling (Schuster and Paliwal, 1997). 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 Palangi et al. (2015), 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:

$$e_t = W_{em} x_t \quad (1)$$

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., (Zeng et al., 2014; Yu et al., 2014).

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:

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

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 (Glorot and Bengio, 2010).

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., (Schuster and Paliwal, 1997). 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:

$$h_t = h_t^{fw} + h_t^{bw} \quad (3)$$

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:

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

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 (Zeng et al., 2014). 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 (Palangi et al., 2015) 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 (Bengio et al., 1994).

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

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

of a sentence can be achieved by merging representations of the local patterns. The max-pooling is formulated as follows:

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

where m 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:

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

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:

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

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 Collobert et al. (2011), and utilizes the stochastic gradient descent (SGD) algorithm. Specifically, the back propagation through time (BPTT) (Werbos, 1990) is employed to compute the gradients layer by layer, and the fan-in technique proposed by Plaut and Hinton (1987) 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 (LeCun et al., 2012).

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, Zeng et al. (2014) 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 and some works have followed this technique(Nguyen and Grishman, 2015; dos Santos et al., 2015). 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(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 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. And in section 4.3, experiment results show that under most circumstances, PI can be more helpful than PF.

4 Experiments

4.1 Database

We use two different datasets. The first one is the dataset provided by SemEval-2010 Task 8. There are 9 *directional* relations and an additional 'other' relation, resulting in 19 relation classes in total.

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	

Table 1: Statistics of KBP37 .

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 (Hendrickx et al., 2009). Both the data and the evaluation tool are publicly available. [2]

The second dataset is a revision of MIML-RE annotation dataset, provided by Gabor Angeli et al. (2014). They use both the 2010 and 2013 KBP official document collections, as well as a July 2013 dump of Wikipedia as the text corpus for annotation. There are 33811 sentences been annotated. To make the dataset more suitable for our task, we made several refinement:

1. First, we add direction to the relation names, such that 'per:employee_of' is splited into two relations 'per:employee_of(e1,e2)' and 'per:employee_of(e2,e1)' except for 'no_relation'. According to description of KBP task,[3] we replace 'org:parents' with 'org:subsidiaries' and replace 'org:member_of' with 'org:member' (by their reverse directions). This leads to 76 relations in the dataset.

2. Then, we statistic the frequency of each relation with two directions separately. And relations with low frequency are discarded so that both directions of each relation occur more than 100 times in the dataset. To better balance the dataset, 80% 'no_relation' sentences are also randomly discarded.

3. After that, dataset are randomly shuffled and then sentences under each relation are all split into three groups, 70% for training, 10% for development, 20% for test. Finally, we remove those sentences in the development and test set whose entity pairs and relation are appeared in a training sentence simultaneously.

In the rest of this paper, we will call the second dataset KBP37 for the sake of simplicity. Statistics and relation types are shown in Table 1. KBP37 contains 18 *directional* relations and an additional 'no_relation' relation, resulting in 37 relation classes. Notice that KBP37 is different from SemEval-2010 Task 8 in several aspects:

- Pairs of nouns in KBP37 are always entity names which are more sparse than SemEval-2010 task 8. And there are more target nouns that own several words instead of one, which is barely unseen in previous dataset.

- Average length of sentences in KBP37 is much longer than Semeval-2010 task 8, which will be discussed in details in section 5.2

- It is not guaranteed that there exists only one relation per data, though in test set only one relation is offered as the answer.

The last aspect may lead to some inconsistency from our task. But since multi-relation data rarely exists, it can be omitted.

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

[3] http://surdeanu.info/kbp2013/TAC_2013_KBP_Slot_Descriptions_1.0.pdf

4.2 Experimental setup

In order to compare with the work by Socher et al. (2012) and Zeng et al. (2014), we use the same word vectors proposed by Turian et al. (2010) (50-dimensional) to initialize the embedding layer in the main experiments. Additionally, to compare with other recent models, additional experiments are also conducted with the word vectors pre-trained by Mikolov et al. (2013) which are 300-dimensional.

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 feature vector m is $M = 800$, and with Mikolov's 300-dimensional word vectors, the best feature dimension is $M =$. The learning rate is set to be 0.01 in both two conditions. For fast convergence, we set learning rate to 0.1 in the first five iterations. And iteration time is 20.

And since we split a development set for KBP37, it is more convenient to tune the hyper-parameters. In the test process, development data is also used to choose the best model among different iterations. For KBP37, we only use Turian's 50-dimensional word vectors. The best dimension of feature vector m is $M = 700$. And max iteration time is 20.

4.3 Results

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

Table 2: F1 results with the proposed RNN model on SemEval-2010 Task 8 dataset, + shows the performance after adding each modification.

Table 2 presents the F1 results of proposed RNN model, with the contribution offered by each modification. It can be seen that the basic RNN, which is signal directional 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 Zeng et al. (2014), where the positional features lead to nearly 10 percentiles of F1 improvement, which is similar as the gain obtained in our model.

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

Table 3: Comparing F1 scores with different neural models, different position usage and different datasets. The 50-dimensional word vectors provided by Turian et al. (2010) are used for pre-training. PF stands for position features and PI stands for position indicators. Number in the parentheses shows the best dimension of hidden layer.

The second experiment compares three representative neural models with different datasets and different position information added. The results are presented in Table 3. The 50-dimensional word vectors are employed in this table. In our experiments, though rather close, we didn't reproduce 78.9 of F1 value reported by Zeng et al. (2014), the third and fourth column shows our results with CNN-based model. Compared among different rows, we can come to the conclusion that the RNN model outperforms both the MV-RNN model proposed by (Socher et al., 2012) and the CNN model proposed by (Zeng et al., 2014). Compared among different columns, the results show that RNN model obtains more improvement comparing to CNN on KBP37 dataset, which also indicates the difference between two

datasets. More discussion will be held in section 5.2.

From Table 3, we can draw another conclusion that, with recurrent structure, PI is more effective than PF. And PI contributes even more when experiments are implemented on KBP37 dataset. The former phenomenon should caused by the ambiguous accumulation in the recurrent process which may disrupt the PF information. The latter probably attributes to a shortcoming of PF that it tends to be less accurate when the nouns become phrases instead of single words. More discussion will be held in Section 5.1

An 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.

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 datasets into 5 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. The position indicator does not count. Clearly, long contexts lead to long-distance patterns. In order to compare performance of the RNN and CNN models, we produce the CNN-based method with PI. This modification ensures that the two models learn the same input sequence with the same representation.

The F1 results on the 5 subsets are reported in Figure 2. It can be seen that 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. The results also shows that when PF is used, there is no such clear trend, which means PF is more suitable for CNN models and PI is more suitable for RNN models. But since PI performs better than PF with CNN in KBP37(51.3 to 55.1 from Table 3), PI could be more robust.

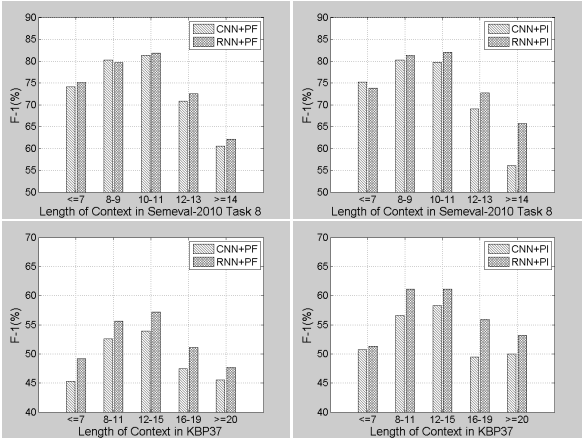

Figure 2: F1 scores with different length of contexts in Semeval-2010 Task 8 and KBP37.

Note that with both 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.

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 3, where the discrepancy between the two models in SemEval-2010 dataset is not so remarkable (77.4 vs. 79.6). This can be attributed to the small proportion of long contexts in test data.

To reflect the limitation of SemEval-2010 dataset, the distribution of the context lengths is calculated on the test dataset. For comparison, the New York Time corpus with the entities and relations selected from a subset of Freebase recommended by Riedel et al. (2013) is also presented.

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 SemEval-2010 task 8 dataset is rather small compared to the other two datasets. This suggests that the strengths of the different models were not fully demonstrated by only implementing experiments on SemEval-2010 task 8 dataset. Since most recent works on relation classification only implemented on this single dataset, a comparison among different models on KBP37 dataset is needed.

Dataset	Context Length			Proportion of Long Context ($\geq$ 11)
	$\leq$10	11 - 15	$\geq$ 16	
SemEval-2010 task-8 (Hendrickx et al., 2009)	6658	3725	334	0.379
NYT+Freebase (Riedel et al., 2013)	22057	19369	3889	0.513
KBP+Wikipedia (Angeli et al., 2014)	6618	11647	15546	0.804

Table 4: The distribution of context lengths with three datasets.

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.

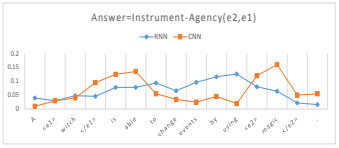

Figure 3: Semantic distribution on words in the sentence "A <e1> witch </e1> is able to change events by using <e2> magic </e2> ."

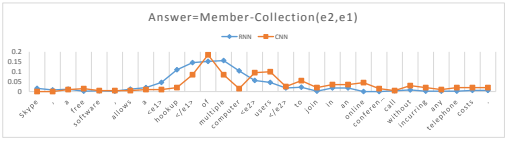

Figure 4: Semantic distribution on words in the sentence "Skype, a free software, allows a <e1> hookup </e1> of multiple computer <e2> users </e2> to join in an online conference call without incurring any telephone costs."

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 simple RNN-based approach for relation classification. Compared to other deep learning models such as 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 on two different datasets

demonstrated that the RNN-based approach can achieve better results than CNN-based approach, and for sentences with long-distance relations, the RNN model exhibits clear advantages.

References

Gabor Angeli, Julie Tibshirani, Jean Y. Wu, and Christopher D. Manning. 2014. Combining distant and partial supervision for relation extraction. In *EMNLP*, October.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Mikael Boden. 2002. A guide to recurrent neural networks and backpropagation. In *In the Dallas project, SICS Technical Report T2002:03*.

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

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 626–634.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. 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*, pages 94–99. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. 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*, pages 541–550. Association for Computational Linguistics.

Nanda Kambhatla. 2004. 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*, page 22. Association for Computational Linguistics.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. 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*, pages 1045–1048.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. 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*, pages 1003–1011. Association for Computational Linguistics.

Raymond J Mooney and Razvan C Bunescu. 2005. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *In Proceedings of NAACL Workshop on Vector Space Modeling for NLP*.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2015. Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. *arXiv preprint arXiv:1502.06922*.

David C Plaut and Geoffrey E Hinton. 1987. Learning sets of filters using back-propagation. *Computer Speech & Language*, 2(1):35–61.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*, pages 74–84.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. 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*, pages 712–717. ACM.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. 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*, pages 455–465. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Mo Yu, Matthew R Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In *NIPS 2014 Workshop*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. 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*, pages 2335–2344.