Relation Classification via Convolutional Deep Neural Network

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou and Jun Zhao
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
95 Zhongguancun East Road, Beijing 100190, China
{djzeng,kliu,swlai,gyzhou,jzhao}@nlpr.ia.ac.cn

Abstract

The state-of-the-art methods used for relation classification are primarily based on statistical machine learning, and their performance strongly depends on the quality of the extracted features. The extracted features are often derived from the output of pre-existing natural language processing (NLP) systems, which leads to the propagation of the errors in the existing tools and hinders the performance of these systems. In this paper, we exploit a convolutional deep neural network (DNN) to extract lexical and sentence level features. Our method takes all of the word tokens as input without complicated pre-processing. First, the word tokens are transformed to vectors by looking up word embeddings[1]. Then, lexical level features are extracted according to the given nouns. Meanwhile, sentence level features are learned using a convolutional approach. These two level features are concatenated to form the final extracted feature vector. Finally, the features are fed into a softmax classifier to predict the relationship between two marked nouns. The experimental results demonstrate that our approach significantly outperforms the state-of-the-art methods.

1 Introduction

The task of relation classification is to predict semantic relations between pairs of nominals and can be defined as follows: given a sentence S with the annotated pairs of nominals e_1 and e_2, we aim to identify the relations between e_1 and e_2 (Hendrickx et al., 2010). There is considerable interest in automatic relation classification, both as an end in itself and as an intermediate step in a variety of NLP applications.

The most representative methods for relation classification use supervised paradigm; such methods have been shown to be effective and yield relatively high performance (Zelenko et al., 2003; Bunescu and Mooney, 2005; Zhou et al., 2005; Mintz et al., 2009). Supervised approaches are further divided into feature-based methods and kernel-based methods. Feature-based methods use a set of features that are selected after performing textual analysis. They convert these features into symbolic IDs, which are then transformed into a vector using a paradigm that is similar to the bag-of-words model[2]. Conversely, kernel-based methods require pre-processed input data in the form of parse trees (such as dependency parse trees). These approaches are effective because they leverage a large body of linguistic knowledge. However, the extracted features or elaborately designed kernels are often derived from the output of pre-existing NLP systems, which leads to the propagation of the errors in the existing tools and hinders the performance of such systems (Bach and Badaskar, 2007). It is attractive to consider extracting features that are as independent from existing NLP tools as possible.

To identify the relations between pairs of nominals, it is necessary to a skillfully combine lexical and sentence level clues from diverse syntactic and semantic structures in a sentence. For example, in the sentence "The $[\text{fire}]_{e_1}$ inside WTC was caused by exploding $[\text{fuel}]_{e_2}$", to identify that *fire* and *fuel* are in a

[1]A word embedding is a distributed representation for a word. For example, Collobert et al. (2011) use a 50-dimensional vector to represent a word.

[2]http://en.wikipedia.org/wiki/Bag-of-words_model

Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 2335–2344, Dublin, Ireland, August 23-29 2014.

Cause-Effect relationship, we usually leverage the marked nouns and the meanings of the entire sentence. In this paper, we exploit a convolutional DNN to extract lexical and sentence level features for relation classification. Our method takes all of the word tokens as input without complicated pre-processing, such as Part-of-Speech (POS) tagging and syntactic parsing. First, all the word tokens are transformed into vectors by looking up word embeddings. Then, lexical level features are extracted according to the given nouns. Meanwhile, sentence level features are learned using a convolutional approach. These two level features are concatenated to form the final extracted feature vector. Finally, the features are feed into a softmax classifier to predict the relationship between two marked nouns.

The idea of extracting features for NLP using convolutional DNN was previously explored by Collobert et al. (2011), in the context of POS tagging, chunking (CHUNK), Named Entity Recognition (NER) and Semantic Role Labeling (SRL). Our work shares similar intuition with that of Collobert et al. (2011). In (Collobert et al., 2011), all of the tasks are considered as the sequential labeling problems in which each word in the input sentence is given a tag. However, our task, "relation classification", can be considered a multi-class classification problem, which results in a different objective function. Moreover, relation classification is defined as assigning relation labels to pairs of words. It is thus necessary to specify which pairs of words to which we expect to assign relation labels. For that purpose, the position features (PF) are exploited to encode the relative distances to the target noun pairs. To the best of our knowledge, this work is the first example of using a convolutional DNN for relation classification.

The contributions of this paper can be summarized as follows.

- We explore the feasibility of performing relation classification without complicated NLP pre-processing. A convolutional DNN is employed to extract lexical and sentence level features.
- To specify pairs of words to which relation labels should be assigned, position features are proposed to encode the relative distances to the target noun pairs in the convolutional DNN.
- We conduct experiments using the SemEval-2010 Task 8 dataset. The experimental results demonstrate that the proposed position features are critical for relation classification. The extracted lexical and sentence level features are effective for relation classification. Our approach outperforms the state-of-the-art methods.

2 Related Work

Relation classification is one of the most important topics in NLP. Many approaches have been explored for relation classification, including unsupervised relation discovery and supervised classification. Researchers have proposed various features to identify the relations between nominals using different methods.

In the unsupervised paradigms, contextual features are used. Distributional hypothesis theory (Harris, 1954) indicates that words that occur in the same context tend to have similar meanings. Accordingly, it is assumed that the pairs of nominals that occur in similar contexts tend to have similar relations. Hasegawa et al. (2004) adopted a hierarchical clustering method to cluster the contexts of nominals and simply selected the most frequent words in the contexts to represent the relation between the nominals. Chen et al. (2005) proposed a novel unsupervised method based on model order selection and discriminative label identification to address this problem.

In the supervised paradigm, relation classification is considered a multi-classification problem, and researchers concentrate on extracting more complex features. Generally, these methods can be categorized into two types: feature-based and kernel-based. In feature-based methods, a diverse set of strategies have been exploited to convert the classification clues (such as sequences and parse trees) into feature vectors (Kambhatla, 2004; Suchanek et al., 2006). Feature-based methods suffer from the problem of selecting a suitable feature set when converting the structured representation into feature vectors. Kernel-based methods provide a natural alternative to exploit rich representations of the input classification clues, such as syntactic parse trees. Kernel-based methods allow the use of a large set of features without explicitly extracting the features. Various kernels, such as the convolution tree kernel (Qian et

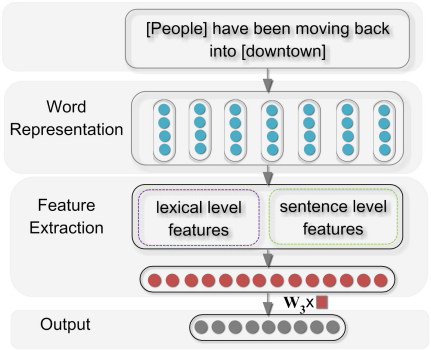

Figure 1: Architecture of the neural network used for relation classification.

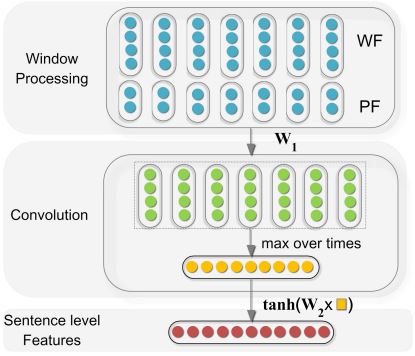

Figure 2: The framework used for extracting sentence level features.

al., 2008), subsequence kernel (Mooney and Bunescu, 2005) and dependency tree kernel (Bunescu and Mooney, 2005), have been proposed to solve the relation classification problem. However, the methods mentioned above suffer from a lack of sufficient labeled data for training. Mintz et al. (2009) proposed distant supervision (DS) to address this problem. The DS method selects sentences that match the facts in a knowledge base as positive examples. The DS algorithm sometimes faces the problem of wrong labels, which results in noisy labeled data. To address the shortcoming of DS, Riedel et al. (2010) and Hoffmann et al. (2011) cast the relaxed DS assumption as multi-instance learning. Furthermore, Takamatsu et al. (2012) noted that the relaxed DS assumption would fail and proposed a novel generative model to model the heuristic labeling process in order to reduce the wrong labels.

The supervised method has been demonstrated to be effective for relation detection and yields relatively high performance. However, the performance of this method strongly depends on the quality of the designed features. With the recent revival of interest in DNN, many researchers have concentrated on using *Deep Learning* to learn features. In NLP, such methods are primarily based on learning a distributed representation for each word, which is also called a word embeddings (Turian et al., 2010). Socher et al. (2012) present a novel recursive neural network (RNN) for relation classification that learns vectors in the syntactic tree path that connects two nominals to determine their semantic relationship. Hashimoto et al. (2013) also use an RNN for relation classification; their method allows for the explicit weighting of important phrases for the target task. As mentioned in Section 1, it is difficult to design high quality features using the existing NLP tools. In this paper, we propose a convolutional DNN to extract lexical and sentence level features for relation classification; our method effectively alleviates the shortcomings of traditional features.

3 Methodology

3.1 The Neural Network Architecture

Figure 1 describes the architecture of the neural network that we use for relation classification. The network takes an input sentence and discovers multiple levels of feature extraction, where higher levels represent more abstract aspects of the inputs. It primarily includes the following three components: *Word Representation*, *Feature Extraction* and *Output*. The system does not need any complicated syntactic or semantic preprocessing, and the input of the system is a sentence with two marked nouns. Then, the word tokens are transformed into vectors by looking up word embeddings. In succession, the lexical and sentence level features are respectively extracted and then directly concatenated to form the final feature vector. Finally, to compute the confidence of each relation, the feature vector is fed into a softmax classifier. The output of the classifier is a vector, the dimension of which is equal to the number of predefined relation types. The value of each dimension is the confidence score of the corresponding relation.

Features	Remark
L1	Noun 1
L2	Noun 2
L3	Left and right tokens of noun 1
L4	Left and right tokens of noun 2
L5	WordNet hypernyms of nouns

Table 1: Lexical level features.

3.2 Word Representation

In the *word representation* component, each input word token is transformed into a vector by looking up word embeddings. Collobert et al. (2011) reported that word embeddings learned from significant amounts of unlabeled data are far more satisfactory than the randomly initialized embeddings. In relation classification, we should first concentrate on learning discriminative word embeddings, which carry more syntactic and semantic information, using significant amounts of unlabeled data. Unfortunately, it usually takes a long time to train the word embeddings[3]. However, there are many trained word embeddings that are freely available (Turian et al., 2010). A comparison of the available word embeddings is beyond the scope of this paper. Our experiments directly utilize the trained embeddings provided by Turian et al.(2010).

3.3 Lexical Level Features

Lexical level features serve as important cues for deciding relations. The traditional lexical level features primarily include the nouns themselves, the types of the pairs of nominals and word sequences between the entities, the quality of which strongly depends on the results of existing NLP tools. Alternatively, this paper uses generic word embeddings as the source of base features. We select the word embeddings of marked nouns and the context tokens. Moreover, the WordNet hypernyms[4] are adopted as MVRNN (Socher et al., 2012). All of these features are concatenated into our lexical level features vector $\mathbf{l}$. Table 1 presents the selected word embeddings that are related to the marked nouns in the sentence.

3.4 Sentence Level Features

As mentioned in section 3.2, all of the tokens are represented as word vectors, which have been demonstrated to correlate well with human judgments of word similarity. Despite their success, single word vector models are severely limited because they do not capture long distance features and semantic compositionality, the important quality of natural language that allows humans to understand the meanings of a longer expression. In this section, we propose a max-pooled convolutional neural network to offer sentence level representation and automatically extract sentence level features. Figure 2 shows the framework for sentence level feature extraction. In the *Window Processing* component, each token is further represented as Word Features (WF) and Position Features (PF) (see section 3.4.1 and 3.4.2). Then, the vector goes through a convolutional component. Finally, we obtain the sentence level features through a non-linear transformation.

3.4.1 Word Features

Distributional hypothesis theory (Harris, 1954) indicates that words that occur in the same context tend to have similar meanings. To capture this characteristic, the WF combines a word's vector representation and the vector representations of the words in its context. Assume that we have the following sequence of words.

$$S : [\text{People}]_0 \quad \text{have}_1 \quad \text{been}_2 \quad \text{moving}_3 \quad \text{back}_4 \quad \text{into}_5 \quad [\text{downtown}]_6$$

The marked nouns are associated with a label y that defines the relation type that the marked pair contains. Each word is also associated with an index into the word embeddings. All of the word tokens of the sentence S are then represented as a list of vectors $(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_6)$, where $\mathbf{x}_i$ corresponds to the word

[3]Collobert et al. (2011) proposed a *pairwise ranking* approach to train the word embeddings, and the total training time for an English corpus (Wikipedia) was approximately four weeks.

[4]http://sourceforge.net/projects/supersensetag/

embedding of the i-th word in the sentence. To use a context size of w, we combine the size w windows of vectors into a richer feature. For example, when we take $w = 3$, the WF of the third word "moving" in the sentence S is expressed as $[\mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}]$. Similarly, considering the whole sentence, the WF can be represented as follows:

$$\{[\mathbf{x}_s, \mathbf{x}_0, \mathbf{x}_1],\ [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2],\ \cdots,\ [\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_e]\}^5$$

3.4.2 Position Features

Relation classification is a very complex task. Traditionally, structure features (e.g., the shortest dependency path between nominals) are used to solve this problem (Bunescu and Mooney, 2005). Apparently, it is not possible to capture such structure information only through WF. It is necessary to specify which input tokens are the target nouns in the sentence. For this purpose, PF are proposed for relation classification. In this paper, the PF is the combination of the relative distances of the current word to w_1 and w_2. For example, the relative distances of "moving" in sentence S to "people" and "downtown" are 3 and -3, respectively. In our method, the relative distances also are mapped to a vector of dimension d_e (a hyperparameter); this vector is randomly initialized. Then, we obtain the distance vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ with respect to the relative distances of the current word to w_1 and w_2, and PF = $[\mathbf{d}_1, \mathbf{d}_2]$. Combining the WF and PF, the word is represented as $[\text{WF}, \text{PF}]^T$, which is subsequently fed into the convolution component of the algorithm.

3.4.3 Convolution

We will see that the word representation approach can capture contextual information through combinations of vectors in a window. However, it only produces local features around each word of the sentence. In relation classification, an input sentence that is marked with target nouns only corresponds to a relation type rather than predicting label for each word. Thus, it might be necessary to utilize all of the local features and predict a relation globally. When using neural network, the convolution approach is a natural method to merge all of the features. Similar to Collobert et al. (2011), we first process the output of *Window Processing* using a linear transformation.

$$\mathbf{Z} = \mathbf{W}_1\mathbf{X} \tag{1}$$

$\mathbf{X} \in \mathbb{R}^{n_0 \times t}$ is the output of the *Window Processing* task, where $n_0 = w \times n$, n (a hyperparameter) is the dimension of feature vector, and t is the token number of the input sentence. $\mathbf{W}_1 \in \mathbb{R}^{n_1 \times n_0}$, where n_1 (a hyperparameter) is the size of hidden layer 1, is the linear transformation matrix. We can see that the features share the same weights across all times, which greatly reduces the number of free parameters to learn. After the linear transformation is applied, the output $\mathbf{Z} \in \mathbb{R}^{n_1 \times t}$ is dependent on t. To determine the most useful feature in the each dimension of the feature vectors, we perform a max operation over time on $\mathbf{Z}$.

$$m_i = \max \mathbf{Z}(i, \cdot) \quad 0 \le i \le n_1 \tag{2}$$

where $\mathbf{Z}(i, \cdot)$ denote the i-th row of matrix $\mathbf{Z}$. Finally, we obtain the feature vector $\mathbf{m} = \{m_1, m_2, \cdots, m_{n_1}\}$, the dimension of which is no longer related to the sentence length.

3.4.4 Sentence Level Feature Vector

To learn more complex features, we designed a non-linear layer and selected hyperbolic tanh as the activation function. One useful property of tanh is that its derivative can be expressed in terms of the function value itself:

$$\frac{d}{dx}\tanh x = 1 - \tanh^2 x \tag{3}$$

It has the advantage of making it easy to compute the gradient in the backpropagation training procedure. Formally, the non-linear transformation can be written as

$$\mathbf{g} = \tanh(\mathbf{W}_2\mathbf{m}) \tag{4}$$

[5] $\mathbf{x}_s$ and $\mathbf{x}_e$ are special word embeddings that correspond to the beginning and end of the sentence, respectively.

$\mathbf{W_2} \in \mathbb{R}^{n_2 \times n_1}$ is the linear transformation matrix, where n_2 (a hyperparameter) is the size of hidden layer 2. Compared with $\mathbf{m} \in \mathbb{R}^{n_1 \times 1}$, $\mathbf{g} \in \mathbb{R}^{n_2 \times 1}$ can be considered higher level features (sentence level features).

3.5 Output

The automatically learned lexical and sentence level features mentioned above are concatenated into a single vector $\mathbf{f} = [\mathbf{l}, \mathbf{g}]$. To compute the confidence of each relation, the feature vector $\mathbf{f} \in \mathbb{R}^{n_3 \times 1}$ (n_3 equals n_2 plus the dimension of the lexical level features) is fed into a softmax classifier.

$$\mathbf{o} = \mathbf{W}_3 \mathbf{f} \tag{5}$$

$\mathbf{W_3} \in \mathbb{R}^{n_4 \times n_3}$ is the transformation matrix and $\mathbf{o} \in \mathbb{R}^{n_4 \times 1}$ is the final output of the network, where n_4 is equal to the number of possible relation types for the relation classification system. Each output can be then interpreted as the confidence score of the corresponding relation. This score can be interpreted as a conditional probability by applying a softmax operation (see Section 3.6).

3.6 Backpropagation Training

The DNN based relation classification method proposed here could be stated as a quintuple $\theta = (\mathbf{X}, \mathbf{N}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3)$[6]. In this paper, each input sentence is considered independently. Given an input example s, the network with parameter θ outputs the vector $\mathbf{o}$, where the i-th component o_i contains the score for relation i. To obtain the conditional probability $p(i|x, \theta)$, we apply a softmax operation over all relation types:

$$p(i|x, \theta) = \frac{e^{o_i}}{\sum_{k=1}^{n_4} e^{o_k}} \tag{6}$$

Given all our (suppose T) training examples $(x^{(i)}; y^{(i)})$, we can then write down the log likelihood of the parameters as follows:

$$J(\theta) = \sum_{i=1}^{T} \log p(y^{(i)}|x^{(i)}, \theta) \tag{7}$$

To compute the network parameter θ, we maximize the log likelihood $J(\theta)$ using a simple optimization technique called stochastic gradient descent (SGD). $\mathbf{N}, \mathbf{W}_1, \mathbf{W}_2$ and $\mathbf{W}_3$ are randomly initialized and $\mathbf{X}$ is initialized using the word embeddings. Because the parameters are in different layers of the neural network, we implement the backpropagation algorithm: the differentiation chain rule is applied through the network until the word embedding layer is reached by iteratively selecting an example (x, y) and applying the following update rule.

$$\theta \leftarrow \theta + \lambda \frac{\partial \log p(y|x, \theta)}{\partial \theta} \tag{8}$$

4 Dataset and Evaluation Metrics

To evaluate the performance of our proposed method, we use the SemEval-2010 Task 8 dataset (Hendrickx et al., 2010). The dataset is freely available[7] and contains 10,717 annotated examples, including 8,000 training instances and 2,717 test instances. There are 9 relationships (with two directions) and an undirected *Other* class. The following are examples of the included relationships: *Cause-Effect*, *Component-Whole* and *Entity-Origin*. In the official evaluation framework, directionality is taken into account. A pair is counted as correct if the order of the words in the relationship is correct. For example, both of the following instances S_1 and S_2 have the relationship *Component-Whole*.

S_1 : The $[\text{haft}]_{e_1}$ of the $[\text{axe}]_{e_2}$ is make $\cdots \Rightarrow$ Component-Whole(e_1,e_2)

S_2 : This $[\text{machine}]_{e_1}$ has two $[\text{units}]_{e_2}$ $\cdots \Rightarrow$ Component-Whole(e_2,e_1)

[6] $\mathbf{N}$ represents the word embeddings of WordNet hypernyms.

[7] http://docs.google.com/View?id=dfvxd49s_36c28v9pmw

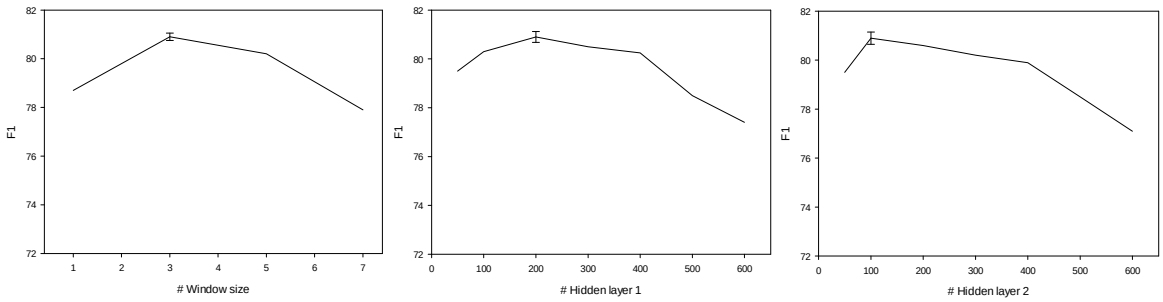

Figure 3: Effect of hyperparameters.

However, these two instances cannot be classified into the same category because *Component-Whole(e_1,e_2)* and *Component-Whole(e_2,e_1)* are different relationships. Furthermore, the official ranking of the participating systems is based on the macro-averaged F1-scores for the nine proper relations (excluding *Other*). To compare our results with those obtained in previous studies, we adopt the macro-averaged F1-score and also account for directionality into account in our following experiments[8].

5 Experiments

In this section, we conduct three sets of experiments. The first is to test several variants via cross-validation to gain some understanding of how the choice of hyperparameters impacts upon the performance. In the second set of experiments, we make comparison of the performance among the convolutional DNN learned features and various traditional features. The goal of the third set of experiments is to evaluate the effectiveness of each extracted feature.

5.1 Parameter Settings

In this section, we experimentally study the effects of the three parameters in our proposed method: the window size in the convolutional component w, the number of hidden layer 1, and the number of hidden layer 2. Because there is no official development dataset, we tuned the hyperparameters by trying different architectures via 5-fold cross-validation.

In Figure 3, we respectively vary the number of hyper parameters w, n_1 and n_2 and compute the F1. We can see that it does not improve the performance when the window size is greater than 3. Moreover, because the size of our training dataset is limited, the network is prone to overfitting, especially when using large hidden layers. From Figure 3, we can see that the parameters have a limited impact on the results when increasing the numbers of both hidden layers 1 and 2. Because the distance dimension has little effect on the result (this is not illustrated in Figure 3), we heuristically choose $d_e = 5$. Finally, the word dimension and learning rate are the same as in Collobert et al. (2011). Table 2 reports all the hyperparameters used in the following experiments.

Hyperparameter	*Window size*	*Word dim.*	*Distance dim.*	*Hidden layer 1*	*Hidden layer 2*	*Learning rate*
Value	$w = 3$	$n = 50$	$d_e = 5$	$n_1 = 200$	$n_2 = 100$	$\lambda = 0.01$

Table 2: Hyperparameters used in our experiments.

5.2 Results of Comparison Experiments

To obtain the final performance of our automatically learned features, we select seven approaches as competitors to be compared with our method in Table 3. The first five competitors are described in Hendrickx et al. (2010), all of which use traditional features and employ SVM or MaxEnt as the classifier. These systems design a series of features and take advantage of a variety of resources (WordNet, ProBank, and FrameNet, for example). *RNN* represents recursive neural networks for relation classification, as

[8]The corpus contains a Perl-based automatic evaluation tool.

Classifier	Feature Sets	F1
SVM	POS, stemming, syntactic patterns	60.1
SVM	word pair, words in between	72.5
SVM	POS, stemming, syntactic patterns, WordNet	74.8
MaxEnt	POS, morphological, noun compound, thesauri, Google n-grams, WordNet	77.6
SVM	POS, prefixes, morphological, WordNet, dependency parse, Levin classed, ProBank, FrameNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
RNN	-	74.8
	POS, NER, WordNet	77.6
MVRNN	-	79.1
	POS, NER, WordNet	82.4
Proposed	word pair, words around word pair, WordNet	**82.7**

Table 3: Classifier, their feature sets and the F1-score for relation classification.

proposed by Socher et al. (2012). This method learns vectors in the syntactic tree path that connect two nominals to determine their semantic relationship. The *MVRNN* model builds a single compositional semantics for the minimal constituent, including both nominals as *RNN* (Socher et al., 2012). It is almost certainly too much to expect a single fixed transformation to be able to capture the meaning combination effects of all natural language operators. Thus, *MVRNN* assigns a matrix to every word and modifies the meanings of other words instead of only considering word embeddings in the recursive procedure.

Table 3 illustrates the macro-averaged F1 measure results for these competing methods along with the resources, features and classifier used by each method. Based on these results, we make the following observations:

(1) Richer feature sets lead to better performance when using traditional features. This improvement can be explained by the need for semantic generalization from training to test data. The quality of traditional features relies on human ingenuity and prior NLP knowledge. It is almost impossible to manually choose the best feature sets.

(2) *RNN* and *MVRNN* contain feature learning procedures; thus, they depend on the syntactic tree used in the recursive procedures. Errors in syntactic parsing inhibit the ability of these methods to learn high quality features. *RNN* cannot achieve a higher performance than the best method that uses traditional features, even when POS, NER and WordNet are added to the training dataset. Compared with *RNN*, the *MVRNN* model can capture the meaning combination effectively and achieve a higher performance.

(3) Our method achieves the best performance among all of the compared methods. We also perform a t-test ($p \leqslant 0.05$), which indicates that our method significantly outperforms all of the compared methods.

5.3 The Effect of Learned Features

	Feature Sets	F1
Lexical	L1	34.7
	+L2	53.1
	+L3	59.4
	+L4	65.9
	+L5	73.3
Sentence	WF	69.7
	+PF	78.9
Combination	all	82.7

Table 4: Score obtained for various sets of features on for the test set. The bottom portion of the table shows the best combination of lexical and sentence level features.

In our method, the network extract lexical and sentence level features. The lexical level features primarily contain five sets of features (L1 to L5). We performed ablation tests on the five sets of features from the lexical part of Table 4 to determine which type of features contributed the most. The results are

presented in Table 4, from which we can observe that our learned lexical level features are effective for relation classification. The F1-score is improved remarkably when new features are added. Similarly, we perform experiment on the sentence level features. The system achieves approximately 9.2% improvements when adding PF. When all of the lexical and sentence level features are combined, we achieve the best result.

6 Conclusion

In this paper, we exploit a convolutional deep neural network (DNN) to extract lexical and sentence level features for relation classification. In the network, position features (PF) are successfully proposed to specify the pairs of nominals to which we expect to assign relation labels. The system obtains a significant improvement when PF are added. The automatically learned features yield excellent results and can replace the elaborately designed features that are based on the outputs of existing NLP tools.

Acknowledgments

This work was sponsored by the National Basic Research Program of China (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61272332, 61333018, 61202329, 61303180). This work was supported in part by Noah's Ark Lab of Huawei Tech. Co. Ltd. We thank the anonymous reviewers for their insightful comments.

References

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. *Literature review for Language and Statistics II*.

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.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2005. Unsupervised feature selection for relation extraction. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 262–267.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 415–422.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó. Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 33–38.

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.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics on Interactive poster and demonstration sessions*.

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*, pages 1003–1011.

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

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 697–704.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, pages 148–163.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

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.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pages 721–729.

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.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

GuoDong Zhou, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 427–434.