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

State-of-the-art methods for relation classification are mostly based on statistical machine learning, and the performance heavily depends on the quality of the extracted features. The extracted features are often derived from the output of pre-existing NLP systems, which lead to the error propagation of existing tools and hinder the performance of the system. In this paper, we exploit a convolutional Deep Neural Network (DNN) to extract lexical and sentence level features. Our method takes all the word tokens as input without complicated pre-processing. First, all the word tokens are transformed to 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 as the final extracted feature vector. Finally, the features are fed into a softmax classifier to predict the relationship between two marked nouns. Experimental results show 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, which can be defined as follows: given a sentence $S$ with pairs of nominals $e_1$ and $e_2$ annotated, we aim to identify the relations between $e_1$ and $e_2$ (Hendrickx et al., 2010). The researchers have a considerable interest in automatic relation classification, both as an end in itself and as an intermediate step in a variety of Natural Language Processing (NLP) applications.

The most representative methods for relation classification use supervised paradigm, which 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 (1) feature based methods and (2) kernel based methods. Feature based methods use a set of features selected after performing textual analysis and convert them to symbolic IDs, which is then transformed into vector using a paradigm similar to Bag-of-words model. On the other hand, kernel methods require pre-processed input data in the form of parse trees (e.g., dependency parse tree etc.). 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 lead to the error propagation of existing tools and hinder the performance of the system (Bach and Badaskar, 2007). It is attractive to consider extracting features which are independent on a existing NLP tools as much as possible.

To identify the relations between pairs of nominals, it needs a skilful combination of lexical and sentence level clues from diverse syntactic and semantic structures in a sentence. For example, in the sentence “The [fire]$_{e_1}$ inside WTC was caused by exploding [fuel]$_{e_2}$”, to identify that fire and fuel are in a Cause-Effect relationship, we usually leverage the marked nouns as well as the meanings of

\[^1\text{This work is licenced under a Creative Commons Attribution 4.0 International License. Page numbers and proceedings footer are added by the organizers. License details: http://creativecommons.org/licenses/by/4.0/}\]

\[^1\text{A distributed representation for each word. For example, Collobert (2011) use a 50 dimension vector to represent a word.}\]

\[^2\text{http://en.wikipedia.org/wiki/Bag-of-words\_model}\]
whole sentence. In this paper, we exploit a convolutional Deep Neural Network (DNN) to extract lexical and sentence level features for relation classification. Our method takes all 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 to 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 as 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 through convolutional DNN in NLP has been 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 Collobert et al. (2011). In (Collobert et al., 2011), all the tasks are considered as the sequential labeling problems (give each word in the input sentence a tag). However, our task “relation classification” can be regarded as a multi-class classification problem, leading to 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 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, it is the first trial that using 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 which pairs of words to assign relation labels, the position features are proposed to encode the relative distances to the target noun pairs in the convolutional DNN.
- We conduct the experiments on the dataset of SemEval-2010 Task 8. Experimental results show 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, such as bootstrapping, unsupervised relation discover and supervised classification. Researchers have proposed various features to identify the relations between two nominals by using different methods.

In bootstrapping and unsupervised paradigms, contextual features are used. The Distributional Hypothesis (Harris, 1954) theory indicates that words occur in the same context tend to have similar meanings. Accordingly, it is assumed that the pairs of nominals occurring 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 that is held between the nominals. Chen et al. (2005) proposed a novel unsupervised method based on model order selection and discriminative label identification to deal with this problem.

In the supervised paradigm, relation classification is considered as 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 (e.g., sequences, parse trees, etc.) 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 methods provide a natural alternative to exploit rich representation of the input classification clues like syntactic parse trees, etc. Kernel methods allow the use of a large set of features without extracting them explicitly. So far, various kernels have been proposed to solve relation classification problem, such as
convolution tree kernel (Qian et al., 2008), subsequence kernel (Bunescu and Mooney, 2006) and dependency tree kernel (Bunescu and Mooney, 2005). The methods mentioned above, however, suffered from lacking of a large amount of labeled data for training. Mintz et al. (2009) proposed Distant Supervision (DS) to address this problem. The DS paradigm selected the sentences that matched the facts in knowledge base as positive examples. DS algorithm sometimes exposed to wrong label problem and brought 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) pointed out 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 paradigm has been shown to be effective for relation detection and yields relatively high performance. However, the performance heavily depends on the quality of the designed features. With the recent revival of interest in DNN, many researchers concentrated on using Deep Learning to learn features. In NLP area, it is mainly based on learning a distributed representation for each word, also called word embeddings (Turian et al., 2010). Socher et al. (2012) present a novel recursive neural network (RNN) for relation classification, which learn vectors in the syntactic tree path connecting two nominals to determine their semantic relationship. Hashimoto et al. (2013) also use RNN for relation classification, which allows for an explicit weighting of important phrases for the target task. As mentioned in Section 1, it is difficult to design the high quality features through an existing NLP tools. In this paper, we propose a convolutional DNN to extract lexical and sentence level features for relation classification, which effectively alleviate the shortcomings of traditional features.

3 Methodology

3.1 The Neural Network Architecture

Figure 1 describes the neural network architecture for relation classification. The network takes the input sentence and discovers multiple levels of feature extraction, with higher levels representing more abstract aspects of the inputs. It mainly includes the following three parts: Word Representation, Feature Extraction and Output. The system does not need any complicated syntactic or semantic preprocessing and the output 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, which are directly concatenated into 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.

3.2 Word Representation

In word representation part, each input word token is transformed into a vector by looking up word embeddings. Collobert et al. (2011) reported that word embeddings learned from lots of unlabeled da-
ta are far more satisfactory than those randomly initialized embeddings. In relation classification, we should firstly concentrate on learning the discriminative word embeddings carrying more syntactic and semantic information from lots of unlabeled data. However, it usually takes much time to train the word embeddings\(^3\). Meanwhile, there have been many trained word embeddings freely available (Turian et al., 2010). While a comparison of all the word embeddings is beyond the scope of this paper, our experiments directly utilize the trained embeddings provided by SENNA.

### 3.3 Lexical Level Features

Lexical level features serve as important cues for deciding relations. The traditional lexical level features mainly include the nouns themselves, the types of the pairs of nominals and word sequence between the entities, the quality of which relies heavily on the results of the 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\(^4\) of \(e_1\) and \(e_2\) are adopted as MVRNN (Socher et al., 2012). All these features are concatenated into our lexical level features vector \(l\). Table 1 shows the selected word embeddings which are related to the marked nouns in the sentence.

### 3.4 Sentence Level Features

As mentioned in section 3.2, all the tokens are represented as word vectors which have been shown to correlate well with human judgments of word similarity. Despite their success, single word vector models are severely limited since they do not capture the 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 part, we propose a max-pooled convolutional neural network to offer sentence level representation and automatically extract sentence level features. Figure 2 shows the framework of sentence level feature extraction. In the Window Processing, each token is further represent as Word Features (WF) and Position Features (PF) (See section 3.4.1 and 3.4.2). Then the vector is through a convolutional part. Finally, we get the sentence level features through a non-linear transformation.

#### 3.4.1 Word Features

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

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

The marked nouns are associated with a label \(y\) defining the relation type that marked pair contains. Each word is also associated with an index into the word embeddings. All the word tokens of the sentence \(S\) are then represented as a list of vectors \(\{x_0, x_1, \cdots, x_6\}\), where \(x_i\) corresponds to the word embedding of \(i\)-th word in the sentence. To use a context size of \(w\), we combine size \(w\) window 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 \([x_2, x_3, x_4]\). Similarly, considering the whole sentence, the WF can be represented as follows:

\[
\{(x_3, x_0, x_1), [x_0, x_1, x_2], \cdots, [x_5, x_6, x_6]\}\]

\(^3\)Collobert et al. (2011) proposed a pairwise ranking approach to train the word embeddings and the total training time was about four weeks on English corpus (Wikipedia).

\(^4\)http://sourceforge.net/projects/supersensetag/

\(^5\)\(x_s\) and \(x_e\) are special word embeddings, corresponding to sentence start and end, respectively.
3.4.2 Position Features

Relation classification is a very complex task. Traditionally, it is usually explore structure features (e.g., the shortest dependency path between nominals) to solve this problem (Bunescu and Mooney, 2005). Apparently, it cannot capture such structure information only through WF. It is necessary to specify which input tokens are target nouns in the sentence. For this purpose, the PF are proposed for relation classification. In this paper, 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 map to a \( d \) (a hyper parameter) dimension vector, which is randomly initialized. Then, we get the distance vector \( d_1 \) and \( d_2 \) with respect to the relative distances of current word to \( w_1 \) and \( w_2 \) and \( PF = [d_1, d_2] \). Combined WF and PF, the word is represented as \([WF, PF]^T\), which is subsequently fed into the convolution part.

3.4.3 Convolution

We will see that the word representation approach can capture the contextual information through the combination of vectors in a window. However, it only produces local features around each word of the sentence. In relation classification, the input sentence that 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 the local features and predict relation globally. When using neural network, convolution approach is a natural way to merge all the features. Similar to Collobert et al. (2011), we first let the output of Window Processing through a linear transformation.

\[
Z = W_1 X
\]

\( X \in \mathbb{R}^{n_0 \times t} \) is the output of Window Processing part, where \( n_0 = w \times n \) and \( n \) (a hyper parameter) is the dimension of feature vector and \( t \) is the token number of the input sentence. \( W_1 \in \mathbb{R}^{n_1 \times n_0} \) is the linear transformation matrix, where \( n_1 \) (a hyper parameter) is the size of hidden layer 1. We can see that the features share the same weights across all times, which greatly reduces the number of free parameters to learn. After linear transformation, the output \( Z \in \mathbb{R}^{n_1 \times t} \) is dependent on \( t \). To find out the most useful feature in the each dimension of the feature vectors, we carry out a max operation over times on \( Z \).

\[
m_i = \max Z(i, \cdot) \quad 0 \leq i \leq n_1
\]

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

3.4.4 Sentence Level Feature Vector

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

\[
\frac{d}{dx} \tanh x = 1 - \tanh^2 x
\]

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

\[
g = \tanh(W_2 m)
\]

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

3.5 Output

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

\[
o = \tanh(W_3 f)
\]
\( W_3 \in \mathbb{R}^{n_4 \times n_3} \) is the transformation matrix and \( o \in \mathbb{R}^{n_4 \times 1} \) is the final output 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 could be stated as a quintuple \( \theta = (X, N, W_1, W_2, W_3) \). In this paper, the input sentence is considered independently. Given an input example \( s \), the network with parameter \( \theta \) output a vector \( o \), for the \( i \)-th component \( o_i \) with respect to the score belongs to relation \( i \). To get the conditional probability \( p(i|x, \theta) \), we apply a softmax operation over all relation types:

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

(6)

Given all our (say \( T \)) training examples \((x(i); y(i))\), we can then write down the log likelihood of the parameters as:

\[
J(\theta) = \sum_{i=1}^{T} \log p(y(i)|x(i); \theta)
\]

(7)

To compute the network parameter \( \theta \), we maximize the log likelihood \( J(\theta) \) using a simple optimization technique called stochastic gradient descent (SGD). \( N, W_1, W_2 \) and \( W_3 \) are randomly initialized and \( X \) is initialized by the word embeddings. Since 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 by iteratively selecting a example \((x, y)\) and applying the following update rule.

\[
\theta \leftarrow \theta + \lambda \frac{\partial \log p(y|x, \theta)}{\partial \theta}
\]

(8)

### 4 Dataset and Evaluation Metrics

To evaluate the performance of our proposed method, we use the dataset of SemEval-2010 Task 8 (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. Among the relationships are: Cause-Effect, Component-Whole, Entity-Origin, etc. In the official evaluation framework, the 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 two instances \( S_1 \) and \( S_2 \) have the relationship Component-Whole.

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

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

However, these two instances cannot classify 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 are based on the macro-averaged F1-scores for the nine proper relations (excluding Other). To compare with former studies, we adopt the macro-averaged F1-score and take the directionality into account as well in our following experiments\(^8\).

---

\(^6\)\( N \) represents the wordembeddings of WordNet hypernyms.

\(^7\)http://docs.google.com/View?id=dfvxd49s_36c28v9pmw

\(^8\)The corpus contains a Perl-based automatic evaluation tool.
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 hyper parameters 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 one is to evaluate the effectiveness of each extracted feature.

5.1 Parameter Settings

In this part, we experimentally study the effect of the three parameters in our proposed method: the window size in the convolutional part $w$, the number of hidden layer 1, and the number of hidden layer 2. Since there is no official development dataset, we tuned the hyper parameters 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, as the size of our training data is limited, the network is prone to overfitting especially when using big hidden layers. From Figure 3, we can see that the parameters have a limited impact on the results when increasing both the number of hidden layer 1 and 2. Since the distance dimension has little effect on the result (not illustrate in Figure 3), we heuristically choose $d_e = 5$. Lastly, the word dimension and the learning rate is the same as SENNA. Table 2 reports all the hyper parameters used in our following experiments.

| Hyper-parameter | 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: Hyper parameters of our experiments.

5.2 Results of Comparison Experiments

| 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        | POS, NER, WorNet                                                             | 74.8 |
| MVRNN      | POS, NER, WorNet                                                             | 77.6 |
| Proposed   | word pair, words around word pair, WorNet                                    | 82.4 |

Table 3: Classifier, their feature sets and F1-score for relation classification.
To get 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 choose SVM or MaxEnt as classifier. These systems design a series of features and take advantage of a variety of resources (WordNet, ProBank, FrameNet, etc.). RNN represents Recursive Neural Networks for relation classification that proposed by Socher et al. (2012). This method learns vectors in the syntactic tree path connecting two nominals to determine their semantic relationship. 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. Henceforth, MVRNN assigns a matrix to every word and modify 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 together with the resources, features and classifier used by each method, from which we have 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 the human ingenuity and prior NLP knowledge. It is almost impossible to choose the best feature sets manually.

2. RNN and MVRNN contain feature learning procedure, they depend on the syntactic tree in the recursive procedure. The errors of syntactic parsing hinder these methods to learn high quality features. RNN cannot achieve a higher performance than the best method that using traditional features even when adding POS, NER and WordNet. Compare to RNN, 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 one-tailed t-test ($p \leq 0.05$) which shows that our method significantly outperforms all the compared methods.

5.3 The Effect of Learned Features

| Feature Sets | F1   |
|--------------|------|
| Lexical      |      |
| L1           | 24.7 |
| +L2          | 53.1 |
| +L3          | 59.4 |
| +L4          | 65.9 |
| +L5          | 73.3 |
| Sentence     |      |
| WF           | 69.7 |
| +PF          | 78.9 |
| Combination  | 82.7 |

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

In our method, the network extract lexical and sentence level features. The lexical level features mainly contain five sets of features (L1 to L5). We ran ablation test on the five sets of features from the lexical part of Table 4 to determine which type of features contributed the most. We evaluated all 32 ($2^5$) combinations of the feature sets. The results are shown in Table 4, from which we can see that our learned lexical level features are effective for relation classification. The F1-score has remarkable improvement when adding new features. Similarly, we do experiment on the sentence level features. The system achieves about 9.2% improvements when adding PF. When combining all the lexical and sentence level features, 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 that we expect to assign relation labels. The system obtains a significant improvement when adding PF. The automatically learned features yield excellent results and can replace the elaborately designed features which 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). 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*, HLT ’05, pages 724–731, Stroudsburg, PA, USA. Association for Computational Linguistics.

Razvan Bunescu and Raymond Mooney. 2006. Subsequence kernels for relation extraction. *Advances in neural information processing systems*, 18:171.

Jinxu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2005. Unsupervised feature selection for relation extraction. In *Proceedings of IJCNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 999888:2493–2537, November.

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, ACL ’04*, Stroudsburg, PA, USA. Association for Computational Linguistics.

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, Seattle, Washington, USA, October. Association for Computational Linguistics.

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, Stroudsburg, PA, USA. 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*, HLT ’11, pages 541–550, Stroudsburg, PA, USA. 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, ACLdemo ’04*, Stroudsburg, PA, USA. Association for Computational Linguistics.

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*, ACL ’09, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

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 (Coling 2008)*, pages 697–704, Manchester, UK, August. Coling 2008 Organizing Committee.
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, ECML PKDD ’10, pages 148–163, Berlin, Heidelberg. Springer-Verlag.

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, EMNLP-CoNLL ’12, pages 1201–1211, Stroudsburg, PA, USA. Association for Computational Linguistics.

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, KDD ’06, pages 712–717, New York, NY, USA. ACM.

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, ACL ’12, pages 721–729, Stroudsburg, PA, USA. 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, ACL ’10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. J. Mach. Learn. Res., 3:1083–1106, March.

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, ACL ’05, pages 427–434, Stroudsburg, PA, USA. Association for Computational Linguistics.