A Position Encoding Convolutional Neural Network Based on Dependency Tree for Relation Classification

Yunlun Yang Yunhai Tong* Shulei Ma Zhi-Hong Deng*
{incomparable-lun, yhtong, mashulei, zhdeng}@pku.edu.cn
Key Laboratory of Machine Perception (Ministry of Education),
School of Electronics Engineering and Computer Science, Peking University,
Beijing 100871, China

Abstract

With the renaissance of neural network in recent years, relation classification has again become a research hotspot in natural language processing, and leveraging parse trees is a common and effective method of tackling this problem. In this work, we offer a new perspective on utilizing syntactic information of dependency parse tree and present a position encoding convolutional neural network (PECNN) based on dependency parse tree for relation classification. First, tree-based position features are proposed to encode the relative positions of words in dependency trees and help enhance the word representations. Then, based on a redefinition of "context", we design two kinds of tree-based convolution kernels for capturing the semantic and structural information provided by dependency trees. Finally, the features extracted by convolution module are fed to a classifier for labelling the semantic relations. Experiments on the benchmark dataset show that PECNN outperforms state-of-the-art approaches. We also compare the effect of different position features and visualize the influence of tree-based position feature by tracing back the convolution process.

1 Introduction

Relation classification focuses on classifying the semantic relations between pairs of marked entities in given sentences (Hendrickx et al., 2010). It is a fundamental task which can serve as a pre-existing system and provide prior knowledge for information extraction, natural language understanding, information retrieval, etc. However, automatic recognition of semantic relation is challenging. Traditional feature based approaches rely heavily on the quantity and quality of hand-crafted features and lexical resources, and it is time-consuming to select an optimal subset of relevant features in order to maximize performance. Though kernel based methods get rid of the feature selection process, they need elaborately designed kernels and are also computationally expensive.

Recently, with the renaissance of neural network, deep learning techniques have been adopted to provide end-to-end solutions for many classic NLP tasks, such as sentence modeling (Socher, 2014; Kim, 2014) and machine translation (Cho et al., 2014). Recursive Neural Network (RNN) (Socher et al., 2012) and Convolutional Neural Network (CNN) (Zeng et al., 2014) have proven powerful in relation classification. In contrast to traditional approaches, neural network based methods own the ability of automatic feature learning and alleviate the problem of severe dependence on human-designed features and kernels.

However, previous researches (Socher et al., 2012) imply that some features exploited by traditional methods are still informative and can help enhance the performance of neural network in relation classification. One simple but effective approach is to concatenate lexical level features to features extracted by neural network and directly pass the combined vector to classifier. In this way, Socher et al. (2012), Liu et al. (2015) achieve better performances when considering some external features produced

*Corresponding authors

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 65–74,
Austin, Texas, November 1-5, 2016.

by existing NLP tools. Another more sophisticated method adjusts the structure of neural network according to the parse trees of input sentences. The results of (Li et al., 2015) empirically suggest syntactic structures from recursive models might offer useful power in relation classification. Besides relation classification, parse tree also gives neural network a big boost in other NLP tasks (Mou et al., 2015; Tai et al., 2015).

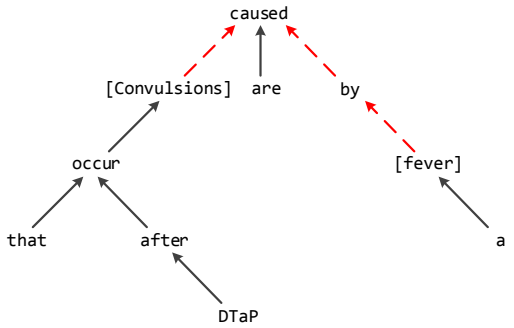

Figure 1: A dependency tree example. Words in square brackets are marked entities. The red dashed-line arrows indicate the path between two entities.

Dependency parse tree is valuable in relation classification task. According to our observation, dependency tree usually shortens the distances between pairs of marked entities and helps trim off redundant words, when comparing with plain text. For example, in the sentence shown in Figure 1, two marked entities span the whole sentence, which brings much noise to the recognition of their relation. By contrast, in the dependency tree corresponding to the sentence, the path between two marked entities comprises only four words and extracts a key phrase "caused by" that clearly implies the relation of entities. This property of dependency tree is ubiquitous and consistent with the Shortest Path Hypothesis which is accepted by previous studies (Bunescu and Mooney, 2005; Xu et al., 2015a; Xu et al., 2015b).

To better utilize the powerful neural network and make the best of the abundant linguistic knowledge provided by parse tree, we propose a position encoding convolutional neural network (PECNN) based on dependency parse tree for relation classification. In our model, to sufficiently benefit from the important property of dependency tree, we introduce the position feature and modify it in the context of parse tree. Tree-based position features encode the relative positions between each word and marked entities in a dependency tree, and help the network pay more attention to the key phrases in sentences. Moreover, with a redefinition of "context", we design two kinds of tree-based convolution kernels for capturing the structural information and salient features of sentences.

To sum up, our contributions are:

1) We propose a novel convolutional neural network with tree-based convolution kernels for relation classification.

2) We confirm the feasibility of transferring the position feature from plain text to dependency tree, and compare the performances of different position features by experiments.

3) Experimental results on the benchmark dataset show that our proposed method outperforms the state-of-the-art approaches. To make the mechanism of our model clear, we also visualize the influence of tree-based position feature on relation classification task.

2 Related Work

Recent studies usually present the task of relation classification in a supervised perspective, and traditional supervised approaches can be divided into feature based methods and kernel methods.

Feature based methods focus on extracting and selecting relevant feature for relation classification. Kambhatla (2004) leverages lexical, syntactic and semantic features, and feeds them to a maximum entropy model. Hendrickx et al. (2010) show that the winner of SemEval-2010 Task 8 used the most types of features and resources, among all participants. Nevertheless, it is difficult to find an optimal feature set, since traversing all combinations of features is time-consuming for feature based methods.

To remedy the problem of feature selection mentioned above, kernel methods represent the input data by computing the structural commonness between sentences, based on carefully designed kernels. Mooney and Bunescu (2005) split sentences into subsequences and compute the similarities using the proposed subsequence kernel. Bunescu and

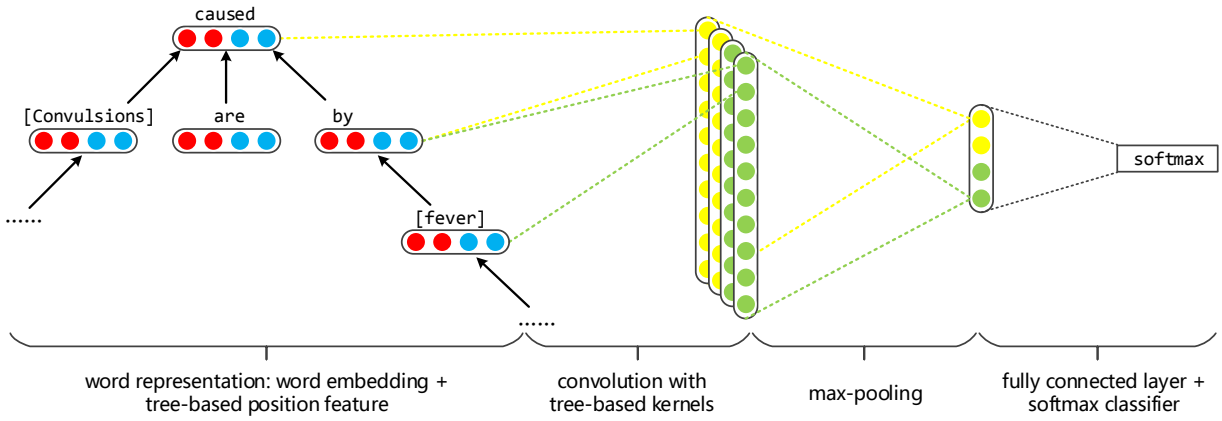

Figure 2: The framework of PECNN. The red and blue circles represent the word embeddings and tree-based position features of words. The yellow and green circles stand for the feature maps extracted by two kinds of convolution kernels respectively.

Mooney (2005) propose a dependency tree kernel and extract information from the Shortest Dependency Path (SDP) between marked entities. Since kernel methods require similarity computation between input samples, they are relatively computationally expensive when facing large-scale datasets.

Nowadays, deep neural network based approaches have become the main solutions to relation classification. Among them, some handle this task by modifying sentence modeling methods. Socher et al. (2012) build RNN on constituency trees of sentences, and apply the model to relation recognition task. Zeng et al. (2014) propose the use of position feature for improving the performance of CNN in relation classification. dos Santos et al. (2015) diminish the impact of noisy class by using a pairwise ranking loss function based CNN. Meanwhile, inspired by the ideas of traditional methods, some recent researches concentrate on mining information from the SDP. Xu et al. (2015b) use a multichannel LSTM network to model the SDP in given sentences. Liu et al. (2015) reserve the subtrees attached to the SDP and propose an augmented SDP based CNN. Neural network based methods offer the advantage of automatic feature learning and also scale well with large amounts of data.

3 Proposed Model

Given a sentence s with two marked entities $e1$ and $e2$, we aim to identify the semantic relation between $e1$ and $e2$ in relation classification. As the set of target relations is predefined, this task can be formulated as a multi-class classification problem. In this section, we detailedly describe our proposed model designed for this problem.

3.1 Framework

The schematic illustration of the framework is shown in Figure 2.

First, the dependency tree of a sentence is generated by the Stanford Parser (Klein and Manning, 2003). For each word in the tree, its word embedding and tree-based position features are concatenated as its representation. The position feature of a word is determined by the relative position between the word and marked entities in the dependency tree.

Next, with tree-based kernels, convolution operations are conducted on each node of the dependency tree. Compared with plain text, dependency tree could provide a word with more meaningful context, thus making tree-based kernel more effective. After convolution, we apply max-pooling over the extracted feature maps to capture the most important features.

At last, the output of max-pooling layer, i.e. the feature vector of input sentence, is fed to a softmax classifier for labelling the semantic relation of entities in each sentence.

3.2 Word Represetation

The representation of a word is composed of two parts: word embedding and tree-based position feature.

3.2.1 Word Embedding

Distributed representation of words in a vector space help learning algorithms to achieve better performance in NLP tasks (Mikolov et al., 2013). Such representation is usually called word embedding in recent works. High-quality word embedding is able to capture precise syntactic and semantic information by training unsupervisedly on large-scale corpora.

In our model, we initialize the word embeddings by pretraining them on a large corpus and further fine-tune them in training phase.

3.2.2 Tree-based Position Feature

Position Feature (PF) is first proposed by (Collobert et al., 2011) for semantic role labeling. (Zeng et al., 2014) exploit position feature as a substitute for traditional structure features in relation classification. The main idea of position feature is to map each discrete distance to a real-valued vector. It is similar to word embedding, except that words are replaced by discrete distances. For instance, let us examine again the sentence shown in Figure 1,

$$[\text{Convulsions}]_{e1} \text{ that occur after DTaP are caused by a } [\text{fever}]_{e2}.$$

the relative distances of *caused* to *Convulsions* and *fever* are respectively 6 and -3. Each relative distance is further mapped to a d_{pf} (a hyperparameter) dimensional vector, which is randomly initialized. Supposing $\boldsymbol{pf}_6$ and $\boldsymbol{pf}_{-3}$ are the corresponding vectors of distance 6 and -3, the position feature of *caused* is given by concatenating these two vectors $[\boldsymbol{pf}_6, \boldsymbol{pf}_{-3}]$.

Position feature on plain text proves to be informative (dos Santos et al., 2015), while it may suffer from several problems. According to our case study, adverbs or unrelated entities that appear between two entities in a sentence could significantly affect the performance of position feature, as these words only change the relative distance to entities without providing any more useful information for relation classification. Similarly, position feature often fails to handle sentences in which marked entities are too far from each other.

On the other hand, dependency tree focuses on the action and agents in a sentence (Socher et al., 2014), which is valuable for relation classification. As we have mentioned above, dependency tree is able to shorten the distances between pairs of marked entities and help trim off redundant words. Therefore, it is straightforward and reasonable to transfer the position feature from plain text to dependency tree.

We propose two kinds of Tree-based Position Feature which we refer as TPF1 and TPF2.

TPF1 encodes the relative distances of current word to marked entities in dependency trees. The "relative distance" here refers to the length of the shortest dependency path between current word and target entity. The sign of the distance is used to distinguish whether current word is a descendant of target entity. After calculating the relative distances of words in the tree, we can get their TPF1 by mapping relative distances to corresponding vectors, which is the same as the PF in plain text.

To more precisely describe the position of a word, TPF2 incorporates more information given by dependency tree. TPF2 represents the relative positions between current word and marked entities by encoding their shortest paths. For a word and an entity, the shortest path between them can be separated by their lowest common ancestor, and the lengths of the two sub-paths are sufficient for encoding the shortest path and the relative position between the word and the entity. As a result, we formally represent the relative position using a 2-tuple, in which two elements are the lengths of the two separated sub-paths respectively. Thereafter, each unique relative position is mapped to a real-valued vector.

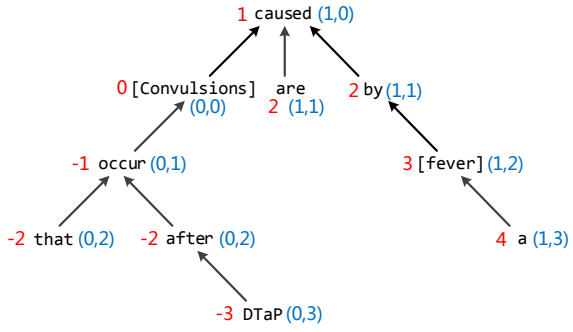

Figure 3: Example of Tree-based Position Features. The red numbers are relative distances in TPF1. The blue 2-tuples are relative positions in TPF2.

For example, in Figure 3, the path between *Convulsions* and *by* is *Convulsions*→ *caused*←*by*. In

TPF1, the relative distance of *by* to *Convulsions* is 2, the length of this path. In TPF2, the lowest common ancestor *caused* splits the path into two subpaths of length 1, so the relative position between *Convulsions* and *by* is $(1, 1)$ (encoded in 2-tuple). More examples of the tree-based position features are shown in Figure 3.

TPF1 and TPF2 both offer good strategies for encoding word position in dependency tree. TPF2 is more fine-grained than TPF1 and TPF1 is a simplified version of TPF2.

In our model, for each word in dependency trees, its word embedding and tree-based position feature are concatenated to form its representation, which is subsequently fed to the convolutional layer.

3.3 Convolution Methods

In the classic CNN architecture of (Collobert et al., 2011) and its variants (Kim, 2014), a convolution window covers a word and its context, i.e. its neighboring words. Thus convolution only captures local features around each word. Words that are not in a same window will not interact, even if they are syntactically related.

Compared with plain text, dependency tree could provide a word with more meaningful context. In a dependency tree, words are connected if they are in some dependency relationship. To capitalize on these syntactic information, we regard the parent and children of a word (i.e. nodes neighboring this word) as its new context. Changing the definition of "context" leads to modification of convolution kernel. To implement this idea, we design two kinds of tree-based kernels (Kernel-1 and Kernel-2), and apply them to sentences in dependency tree form.

Formally, for a word x in the dependency tree, let p be its parent and $c_1, \cdots, c_n$ be its n children. Their vector representation are respectively denoted by $\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{c}_1, \cdots, \boldsymbol{c}_n \in \mathbb{R}^d$. The convolution process of Kernel-1 is formulated as

$$\begin{aligned} \boldsymbol{z}^1_{xi} =& g(W^1_x \cdot \boldsymbol{x} + W^1_p \cdot \boldsymbol{p} + W^1_c \cdot \boldsymbol{c}_i) \\ & for \quad i = 1, \cdots, n \end{aligned} \tag{1}$$

where $\boldsymbol{z}^1_{xi} \in \mathbb{R}^{n_1}$ and n_1 is the number of Kernel-1, and $W^1_x, W^1_p, W^1_c \in \mathbb{R}^{n_1 \times d}$ are weight parameters corresponding to the word, its parent and children respectively. g is the non-linear activation function. For leaf nodes which have no child, i.e. $n = 0$, we assign each of them a child of which the vector representation is $\mathbf{0}$. For the root node, $\boldsymbol{p}$ is set to be $\mathbf{0}$.

Similarly, the output of Kernel-2 is given by

$$\begin{aligned} \boldsymbol{z}^2_{xi} =& g(W^2_x \cdot \boldsymbol{x} + W^2_{lc} \cdot \boldsymbol{c}_i + W^2_{rc} \cdot \boldsymbol{c}_{i+1}) \\ & for \quad i = 1, \cdots, n-1 \end{aligned} \tag{2}$$

where $\boldsymbol{z}^2_{xi} \in \mathbb{R}^{n_2}$ and n_2 is the number of Kernel-2, and $W^2_x, W^2_{lc}, W^2_{rc} \in \mathbb{R}^{n_2 \times d}$ are weight parameters associated with the word and its two neighboring children. If $n \leq 1$, we simply add one or two $\mathbf{0}$ children, just like the zero padding strategy.

Kernel-1 aims at extracting features from words of multiple levels in dependency tree, while Kernel-2 focuses on mining the semantic information between words which share the same parent. Kernel-1 and Kernel-2 both consider 3 words at a time because the experimental results of previous researches (Zeng et al., 2014; dos Santos et al., 2015) suggest that trigram features are relatively more useful in relation classification. And it is also straightforward to extend these kernels to a larger size and apply them to other tasks.

After convolution with tree-based kernels, we apply a global max-pooling operation over extracted features by taking the maximum value in each dimension, which is formulated as

$$\boldsymbol{h}^1 = \operatorname*{elemax}_{x,i} \boldsymbol{z}^1_{xi} \tag{3}$$

$$\boldsymbol{h}^2 = \operatorname*{elemax}_{x,i} \boldsymbol{z}^2_{xi} \tag{4}$$

where $\boldsymbol{h}^1 \in \mathbb{R}^{n_1}$, $\boldsymbol{h}^2 \in \mathbb{R}^{n_2}$, and elemax is the operation which gives the element-wise maximum of all input vectors. As a consequence, the output of convolution process is $[\boldsymbol{h}^1, \boldsymbol{h}^2]$, the combination of features extracted by two kinds of kernels.

3.4 Output and Training Objective

After convolution, the extracted feature is further passed to a fully connected softmax layer whose output is the probability distribution over all types of relations.

Since we treat the relation classification task as a multi-class classification problem, the training objective is the cross-entropy error. For regularization, we apply dropout (Srivastava et al., 2014) to the feature vector extracted by convolution and penalize the fully connected layer with l_2 regularizer as well.

Some other dependency tree based methods like (Liu et al., 2015), (Xu et al., 2015a) and (Xu et al., 2015b), all focus on using different kinds of neural networks to model the shortest dependency path (SDP) between entities. By contrast, PECNN extracts features from the whole dependency tree, so that the information out of SDP will be taken into consideration as well. The empirical results of (dos Santos et al., 2015) suggest that when position features exist, modeling the full sentence yields a better performance than only using the subsentence between entities. With the help of tree-based position feature, our model is capable of evaluating the importance of different parts of dependency trees and tends to pay relatively more attention to SDP.

Some methods enhancing their performances by proposing dataset-specific strategies. dos Santos et al. (2015) treat the class *Other* as a special class and omit its embedding. Xu et al. (2015a) take the relation dimensionality into account and introduce a negative sampling strategy to double the number of training samples, which can be regarded as data augmentation. These strategies do not conflict with our model, but we decide not to integrate them into our methods as we aim to offer a general and effective feature extraction model for relation classification.

4 Experiments

4.1 Dataset and Evaluation Metric

To evaluate our method, we conduct experiments on the SemEval2010 Task 8 dataset which is a widely used benchmark for relation classification. The dataset contains $8,000$ training sentences and $2,717$ test sentences. In each sentence, two entities are marked as target entities.

The predefined target relations include 9 directed relations and an undirected *Other* class. The 9 directed relations are *Cause-Effect, Component-Whole*, *Content-Container*, *Entity- Destination*, *Entity-Origin*, *Instrument-Agency*, *Member-Collection*, *Message-Topic* and *Product-Producer*. "Directed" here means, for example, *Cause-Effect*(e_1, e_2) and *Cause-Effect*(e_2, e_1) are two different relations. In another word, the directionality of relation also matters. And sentences that do not belong to any directed relation are labelled as *Other*. Therefore, relation classification on this dataset is a 19-class classification problem.

Following previous studies, we use the official evaluation metric, macro-averaged F1-score with directionality taken into account and the *Other* class ignored.

4.2 Training Details

Since there is no official validation set, 10% of the training sentences are taken out for hyperparameter tuning and early stopping.

When converting sentences to dependency trees, we note that some prepositions such as "by", "in" and "of", might be important clues to relation classification. To reserve these valuable information, we use the Stanford Parser **without** the *collapsed* option.

In the dataset, there are some entities consisting of multiple words, which make the calculation of relative position ambiguous. To solve this problem, we take the last word as the representation of an entity, as the last word is usually the predominant word.

For word embeddings, we initialize them using the 300-dimensional `word2vec` vectors[1]. The vectors are trained on 100 billion words from Google News. Words not present in the `word2vec` vectors are initialized by sampling each dimension from a uniform distribution (Kim, 2014). Tree-based position features are 50-dimensional and initialized randomly. Therefore the representation of each word has dimensionality of 400.

We use ReLU as the activation function. The number of convolution kernels is 500 for each kind, $1,000$ in total. The dropout rate is 0.5, and the coefficient of l_2 penalty of fully connected layer is set to 10^{-6}. These parameters are selected through grid search on validation set. The network is trained with the Adadelta update rule (Zeiler, 2012). The network is implemented with Theano (Theano Development Team, 2016).

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

Classifier	Features	$F1$
Without External Lexical Features		
MVRNN	word embedding, constituency tree	79.1
CNN	word embedding, position feature	78.9
CR-CNN	word embedding	82.8*
	word embedding, position feature	84.1*
depLCNN	word embedding, dependency tree	81.9
	word embedding, dependency tree	84.0°
SDP-LSTM	word embedding, dependency tree	83.0
PECNN	word embedding, dependency tree, tree-based position feature	**84.0**
With External Lexical Features		
SVM	POS, prefixes, morphological, WordNet, dependency parse Levin classes, PropBankFrameNet, NomLex-Plus, Google n-gram paraphrases, TextRunner	82.2
MVRNN	word embedding, constituency tree, POS, NER, WordNet	82.4
CNN	word embedding, position feature, WordNet	82.7
DepNN	word embedding, dependency tree, WordNet	83.0
	word embedding, dependency tree, NER	83.6
depLCNN	word embedding, dependency tree, WordNet	83.7
	word embedding, dependency tree, WordNet	85.6°
SDP-LSTM	word embedding, dependency tree, POS embedding WordNet embedding, grammar relation embedding	83.7
PECNN	word embedding, dependency tree, tree-based position feature, POS NER, WordNet	**84.6**

Table 1: Comparison of different relation classification models. The symbol * indicates the results with special treatment of the class *Other*. The symbol ° indicates the results with data augmentation strategy.

4.3 Results

The performances of our proposed model and other state-of-the-art methods are shown in Table 1.

First, we compare PECNN with the following baselines when no external lexical feature is used.

Socher et al. (2012) assign a vector and a matrix to each word for the purpose of semantic composition, and build recursive neural network along constituency tree (MVRNN). It is noteworthy that this work is the first one who confirms the feasibility of applying neural network to relation classification.

Following the ideas of (Collobert et al., 2011), Zeng et al. (2014) first solve relation classification using convolutional neural network (CNN). The position feature introduced by them proves effective. dos Santos et al. (2015) build a similar CNN called CR-CNN but replace the objective function with a pairwise ranking loss. By treating the noisy class *Other* as a **special** class, this method achieves an $F1$ of 84.1. The $F1$ score is 82.7 if no special treatment is applied.

The rest two baselines focus on modeling the Shortest Dependency Paths (SDP) between marked entities. Xu et al. (2015a)) (depLCNN) integrate the relation directionality into CNN and achieve an $F1$ of 84.0 with a data augmentation strategy called negative sampling. Without such data augmentation, their $F1$ score is 81.9. Xu et al. (2015b) (SDP-LSTM) represent heterogeneous features as embeddings and propose a multichannel LSTM based recurrent neural network for picking up information along the SDP. Their $F1$ score is 83.0 when only word embedding is used as the word representation.

Without considering any external lexical feature and dataset-specific strategy, our model achieve an $F1$ of 84.0, suggesting that tree-based position features and kernels are effective. Comparing with the CNN based on plain text, our model benefits from dependency tree based network and obtain a notable

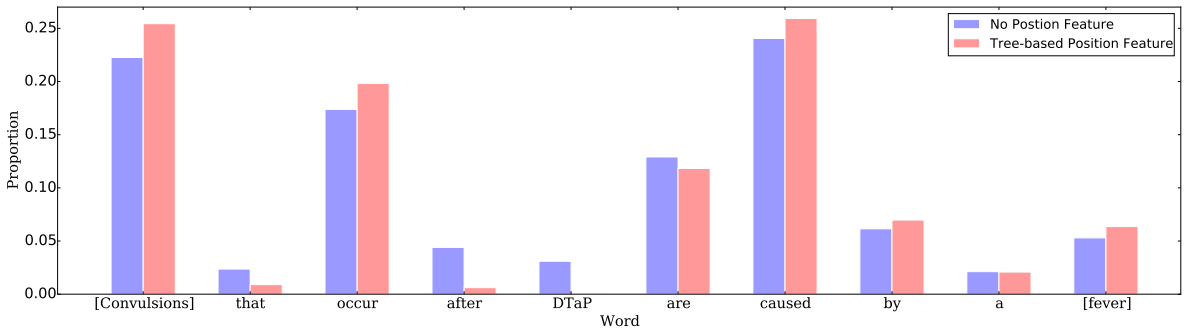

Figure 4: Visualization of the effect of tree-based position feature. The proportions of words change with the use of tree-based position feature.

improvement.

When external lexical features are available, we take two more baselines into account. The first one (SVM) is a typical example of traditional feature-based methods which rely largely on hand-crafted features. Benefitting from various features and resources, this method won the SemEval 2010 Task 8 by a large margin (Hendrickx et al., 2010). Liu et al. (2015) (DepNN) reserve the subtrees attached to the SDP and propose an augmented SDP based CNN.

Most of these baselines concatenate external lexical features to features extracted by neural network and directly pass the combined vector to classifier. SDP-LSTM represents lexical features as embeddings and enhances its word representation. For fair comparison, we add three features (POS tags, NER tags and WordNet hypernyms of marked entities) to the vector extracted by our model and retrain the network. Thus, our model achieves an $F1$ of 84.6 and outperforms all existing baselines in a fair condition where no data augmentation strategy is adopted. The enhancement we gain from external features is less, comparing with other baselines. This implies that our model is able to mine useful features from limited resources, even no extra information is available.

4.4 Effect of Different Position Features

Position Feature	$F1$
plain text PF	83.21
TPF1	83.99
TPF2	83.90

Table 2: Comparison of different position features.

Table 2 summarizes the performances of proposed model when different position features are exploited. To concentrate on studying the effect of position features, we do not involve lexical features in this section. As the table shows, the position feature on plain text is still effective in our model and we accredit its satisfactory result to the dependency information and tree-based kernels. The $F1$ scores of tree-based position features are higher since they are "specially designed" for our model.

Contrary to our expectation, the more fine-grained TPF2 does not yield a better performance than TPF1, and two kinds of TPF give fairly close results. One possible reason is that the influence of a more elaborated definition of relative position is minimal. As most sentences in this dataset are of short length and their dependency trees are not so complicated, replacing TPF1 with TPF2 usually brings little new structural information and thus results in a similar $F1$ score.

However, though the performances of different position features are close, tree-based position feature is an essential part of our model. The $F1$ score is severely reduced to 75.22 when we remove the tree-based position feature in PECNN.

4.5 Effect of Tree-based Position Feature

For shallow CNN in NLP, visualization offers clear and convincing explanations for the mechanism of neural networks (dos Santos and Gatti, 2014; Mou et al., 2015). Moreover, it is easy to implement.

Note that in the max-pooling step, for each kernel, we select the feature which has the largest value. This feature corresponds to 3 words in the convolu-

tion step, and we regard them as the most relevant words extracted by this kernel, with respect to the sentence . Since there are $1,000$ kernels in total, we count $3,000$ words ($\mathbf{0}$ will be ignored) and calculate the proportion of each different word. Intuitively, the more important a word is in this task, the larger its proportion will be.

In Figure 4, we compare the proportions of words in the example sentence when tree-based position feature (TPF) is used and not. As we can see, the proportions of two entities, *Convulsions* and *fever*, and the phrase *caused by* all increase visibly with the presence of TPF, suggesting that TPF is effective in helping the neural network pay more attention to the crucial words and phrases in a sentence. The word *occur* is also picked up by our model since it is an important candidate clue to relation classification. Meanwhile, the influence of irrelevant entity *DTaP* is remarkably diminished as expected.

5 Conclusion

This work presents a dependency parse tree based convolutional neural network for relation classification. We propose tree-based position features to encode the relative positions of words in a dependency tree. Meanwhile, tree-based convolution kernels are designed to gather semantic and syntactic information in dependency trees. Experimental results prove the effectiveness of our model. Comparing with plain text based CNN, our proposed kernels and position features boost the performance of network by utilizing dependency trees in a new perspective.

6 Acknowledgements

This work is partially supported by the National High Technology Research and Development Program of China (Grant No. 2015AA015403) and the National Natural Science Foundation of China (Grant No. 61170091). We would also like to thank the anonymous reviewers for their helpful comments.

References

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.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

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.

Cıcero Nogueira dos Santos and Maıra Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland*.

Cícero 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 and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 626–634.

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*, pages 33–38. 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.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computa-*

tional Linguistics-Volume 1, pages 423–430. Association for Computational Linguistics.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard H. Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2304–2314.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 285–290.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

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

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2315–2325.

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. Association for Computational Linguistics.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Richard Socher. 2014. *Recursive Deep Learning for Natural Language Processing and Computer Vision*. Ph.D. thesis, Stanford University.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

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, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 536–540.

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, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1785–1794.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.