# A Dependency-Based Neural Network for Relation Classification

**Yang Liu**$^{1,2*}$ **Furu Wei**$^{3}$ **Sujian Li**$^{1,2}$ **Heng Ji**$^{4}$ **Ming Zhou**$^{3}$ **Houfeng Wang**$^{1,2}$
[1]Key Laboratory of Computational Linguistics, Peking University, MOE, China
[2]Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, China
[3]Microsoft Research, Beijing, China
[4]Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY, USA
{cs-ly, lisujian, wanghf}@pku.edu.cn
{furu, mingzhou}@microsoft.com jih@rpi.edu

## Abstract

Previous research on relation classification has verified the effectiveness of using dependency shortest paths or subtrees. In this paper, we further explore how to make full use of the combination of these dependency information. We first propose a new structure, termed augmented dependency path (ADP), which is composed of the shortest dependency path between two entities and the subtrees attached to the shortest path. To exploit the semantic representation behind the ADP structure, we develop dependency-based neural networks (DepNN): a recursive neural network designed to model the subtrees, and a convolutional neural network to capture the most important features on the shortest path. Experiments on the SemEval-2010 dataset show that our proposed method achieves state-of-art results.

## 1 Introduction

Relation classification aims to classify the semantic relations between two entities in a sentence. It plays a vital role in robust knowledge extraction from unstructured texts and serves as an intermediate step in a variety of natural language processing applications. Most existing approaches follow the machine learning based framework and focus on designing effective features to obtain better classification performance.

The effectiveness of using dependency relations between entities for relation classification has been reported in previous approaches (Bach and Badaskar, 2007). For example, Suchanek et al. (2006) carefully selected a set of features from tokenization and dependency parsing, and extended some of them to generate high order features in different ways. Culotta and Sorensen (2004) designed a dependency tree kernel and attached more information including Part-of-Speech tag, chunking tag of each node in the tree. Interestingly, Bunescu and Mooney (2005) provided an important insight that the shortest path between two entities in a dependency graph concentrates most of the information for identifying the relation between them. Nguyen et al. (2007) developed these ideas by analyzing multiple subtrees with the guidance of pre-extracted keywords. Previous work showed that the most useful dependency information in relation classification includes the shortest dependency path and dependency subtrees. These two kinds of information serve different functions and their collaboration can boost the performance of relation classification (see Section 2 for detailed examples). However, how to uniformly and efficiently combine these two components is still an open problem. In this paper, we propose a novel structure named Augmented Dependency Path (ADP) which attaches dependency subtrees to words on a shortest dependency path and focus on exploring the semantic representation behind the ADP structure.

Recently, deep learning techniques have been widely used in exploring semantic representations behind complex structures. This provides us an opportunity to model the ADP structure in a neural network framework. Thus, we propose a dependency-based framework where two neural networks are used to model shortest dependency paths and dependency subtrees separately. One convolutional neural network (CNN) is applied over the shortest dependency path, because CNN is suitable for capturing the most useful features in a flat structure. A recursive neural network (RNN) is used for extracting semantic representations from the dependency subtrees, since RNN is good at modeling hierarchical structures. To connect these two networks, each word on the shortest

[*]Contribution during internship at Microsoft Research.

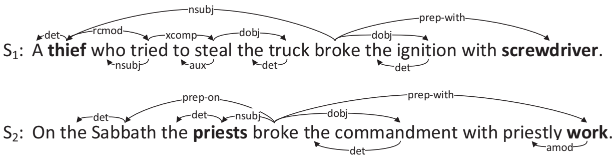
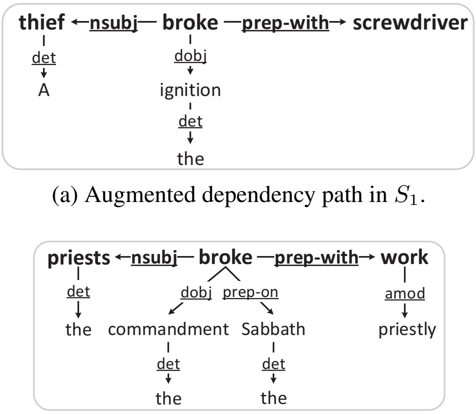

nsubj
prep-with
det
rcmod
xcomp
dobj
dobj
S1: A thief who tried to steal the truck broke the ignition with screwdriver.
nsubj
aux
det
det
prep-on
det
det
nsubj
prep-with
dobj
S2: On the Sabbath the priests broke the commandment with priestly work.
det
amod

Figure 1: Sentences and their dependency trees.


thief ←nsubj— broke —prep-with→ screwdriver
det
A
dobj
ignition
det
the
priests ←nsubj— broke —prep-with→ work
det
the
dobj
prep-on
commandment
Sabbath
det
the
det
the
amod
priestly

(a) Augmented dependency path in $S_1$.

(b) Augmented dependency path in $S_2$.

Figure 2: Augmented dependency paths.

path is combined with a representation generated from its subtree, strengthening the semantic representation of the shortest path. In this way, the augmented dependency path is represented as a continuous semantic vector which can be further used for relation classification.

## 2 Problem Definition and Motivation

The task of relation classification can be defined as follows. Given a sentence $S$ with a pair of entities $e_1$ and $e_2$ annotated, the task is to identify the semantic relation between $e_1$ and $e_2$ in accordance with a set of predefined relation classes (e.g., *Content-Container, Cause-Effect*). For example, in Figure 2, the relation between two entities $e_1$=*thief* and $e_2$=*screwdriver* is *Instrument-Agency*.

Bunescu and Mooney (2005) first used shortest dependency paths between two entities to capture the predicate-argument sequences (e.g., "thief←broke→screwdriver" in Figure 2), which provide strong evidence for relation classification. As we observe, the shortest paths contain more information and the subtrees attached to each node on the shortest path are not exploited enough. For example, Figure 2a and 2b show two instances which have similar shortest dependency paths but belong to different relation classes. Methods only using the path will fail in this case. However, we can distinguish these two paths by virtue of the attached subtrees such as "dobj→commandment" and "dobj→ignition". Based on many observations like this, we propose the idea that combines the subtrees and the shortest path to form a more precise structure for classifying relations. This combined structure is called "**augmented dependency path** (ADP)", as illustrated in Figure 2.

Next, our goal is to capture the semantic representation of the ADP structure between two entities. We first adopt a recursive neural network to model each word according to its attached dependency subtree. Based on the semantic information of each word, we design a convolutional neural network to obtain salient semantic features on the shortest dependency path.

## 3 Dependency-Based Neural Networks

In this section, we will introduce how we use neural network techniques and dependency information to explore the semantic connection between two entities. We dub our architecture of modeling ADP structures as dependency-based neural networks (DepNN). Figure 3 illustrates DepNN with a concrete example. First, we associate each word $w$ and dependency relation $r$ with a vector representation $\boldsymbol{x}_w, \boldsymbol{x}_r \in \mathbb{R}^{dim}$. For each word $w$ on the shortest dependency path, we develop an RNN from its leaf words up to the root to generate a subtree embedding $\boldsymbol{c}_w$ and concatenate $\boldsymbol{c}_w$ with $\boldsymbol{x}_w$ to serve as the final representation of $w$. Next, a CNN is designed to model the shortest dependency path based on the representation of its words and relations. Finally our framework can efficiently represent the semantic connection between two entities with consideration of more comprehensive dependency information.

### 3.1 Modeling Dependency Subtree

The goal of modeling dependency subtrees is to find an appropriate representation for the words on the shortest path. We assume that each word $w$ can be interpreted by itself and its children on the dependency subtree. Then, for each word $w$ on the subtree, its word embedding $\boldsymbol{x}_w \in \mathbb{R}^{dim}$ and subtree representation $\boldsymbol{c}_w \in \mathbb{R}^{dim_c}$ are concatenated to form its final representation $\boldsymbol{p}_w \in \mathbb{R}^{dim+dim_c}$. For a word that does not have a subtree, we set its subtree representation as $\boldsymbol{c}_{LEAF}$. The subtree representation of a word is derived through transforming the representations of its children words.

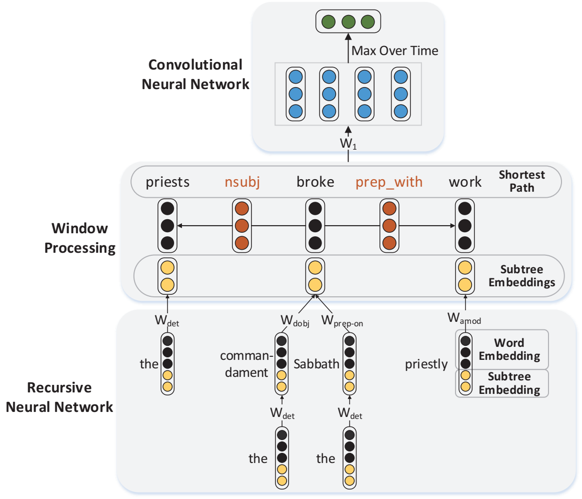


Figure 3: Illustration of Dependency-based Neural Networks.

During the bottom-up construction of the subtree, each word is associated with a dependency relation such as *dobj* as in Figure 3. For each dependency relation $r$, we set a transformation matrix $\boldsymbol{W}_r \in \mathbb{R}^{dim_c \times (dim+dim_c)}$ which is learned during training. Then we can get,

$$\boldsymbol{c}_w = f(\sum_{q \in Children(w)} \boldsymbol{W}_{R_{(w,q)}} \cdot \boldsymbol{p}_q + \boldsymbol{b}) \qquad (1)$$

$$\boldsymbol{p}_q = [\boldsymbol{x}_q, \boldsymbol{c}_q] \qquad (2)$$

where $R_{(w,q)}$ denotes the dependency relation between word $w$ and its child word $q$. This process continues recursively up to the root word on the shortest path.

### 3.2 Modeling Shortest Dependency Path

To classify the semantic relation between two entities, we further explore the semantic representation behind their shortest dependency path, which can be seen as a sequence of words and dependency relations as the bold-font part in Figure 2. As the convolutional neural network (CNN) is good at capturing the salient features from a sequence of objects, we design a CNN to tackle the shortest dependency path.

A CNN contains a convolution operation over a window of object representations, followed by a pooling operation. As we know, a word $w$ on the shortest path is associated with the representation $p_w$ through modeling the subtree. For a dependency relation $r$ on the shortest path, we set its representation as a vector $x_r \in \mathbb{R}^{dim}$. As a sliding window is applied on the sequence, we set the window size as $k$. For example, when $k = 3$, the sliding windows of a shortest dependency path with $n$ words are: $\{[r_s\ w_1\ r_1], [r_1\ w_2\ r_2], \ldots, [r_{n-1}\ w_n\ r_e]\}$ where $r_s$ and $r_e$ are used to denote the beginning and end of a shortest dependency path between two entities.

We concatenate $k$ neighboring words (or dependency relations) representations into a new vector. Assume $\boldsymbol{X}_i \in \mathbb{R}^{dim \cdot k + dim_c \cdot n_w}$ as the concatenated representation of the $i$-th window, where $n_w$ is the number of words in one window. A convolution operation involves a filter $\mathbf{W}_1 \in \mathbb{R}^{l \times (dim \cdot k + dim_c \cdot n_w)}$, which operates on $\boldsymbol{X}_i$ to produce a new feature vector $\boldsymbol{L}_i$ with $l$ dimensions,

$$\boldsymbol{L}_i = \boldsymbol{W}_1 \boldsymbol{X}_i \qquad (3)$$

where the bias term is ignored for simplicity.

Then $\mathbf{W}_1$ is applied to each possible window in the shortest dependency path to produce a feature map: $[L_0, L_1, L_2, \cdots]$. Next, we adopt the widely-used max-over-time pooling operation (Collobert et al., 2011), which can retain the most important features, to obtain the final representation $\boldsymbol{L}$ from the feature map. That is, $\boldsymbol{L} = max(\boldsymbol{L}_0, \boldsymbol{L}_1, \boldsymbol{L}_2, \ldots)$.

### 3.3 Learning

Like other relation classification systems, we also incorporate some lexical level features such as named entity tags and WordNet hypernyms, which prove useful to this task. We concatenate them with the ADP representation $\boldsymbol{L}$ to produce a combined vector $\boldsymbol{M}$. We then pass $\boldsymbol{M}$ to a fully connected $softmax$ layer whose output is the probability distribution $\boldsymbol{y}$ over relation labels.

$$\boldsymbol{M} = [\boldsymbol{L}, \boldsymbol{LEX}] \qquad (4)$$

$$\boldsymbol{y} = softmax(\boldsymbol{W}_2 \boldsymbol{M} + \boldsymbol{b}_2) \qquad (5)$$

Then, the optimization objective is to minimize the cross-entropy error between the ground-truth label vector and the $softmax$ output. Parameters are learned using the back-propagation method (Rumelhart et al., 1988).

## 4 Experiments

We compare DepNN against multiple baselines on SemEval-2010 dataset (Hendrickx et al., 2010).

The training set includes 8000 sentences, and the test set includes 2717 sentences. There are 9 relation types, and each type has two directions. Instances which don't fall in any of these classes are labeled as *Other*. The official evaluation metric is the macro-averaged F1-score (excluding *Other*) and the direction is considered. We use dependency trees generated by the Stanford Parser (Klein and Manning, 2003) with the *collapsed* option.

### 4.1 Contributions of different components

We first show the contributions from different components of DepNN. Two different kinds of word embeddings for initialization are used in the experiments. One is the 50-*d* embeddings provided by SENNA (Collobert et al., 2011). The second is the 200-*d* embeddings used in (Yu et al., 2014), trained on Gigaword with word2vec[1]. All the hyperparameters are set with 5-fold cross-validation.

| Model | F1 | |
|---|---|---|
| | 50-*d* | 200-*d* |
| baseline (Path words) | 73.8 | 75.5 |
| +Depedency relations | 80.3 | 81.8 |
| +Attached subtrees | 81.2 | 82.8 |
| +Lexical features | 82.7 | 83.6 |

Table 1: Performance of DepNN with different components.

We start with a baseline model using a CNN with only the words on the shortest path. We then add dependency relations and attached subtrees. The results indicate that both parts are effective for relation classification. The rich linguistic information embedded in the dependency relations and subtrees can on one hand, help distinguish different functions of the same word, and on the other hand infer an unseen word's role in the sentence. Finally, the lexical features are added and DepNN achieves state-of-the-art results.

### 4.2 Comparison with Baselines

In this subsection, we compare DepNN with several baseline relation classification approaches. Here, DepNN and the baselines are all based on the 200-*d* embeddings trained on Gigaword due to the larger corpus and higher dimensions.

**SVM** (Rink and Harabagiu, 2010): This is the top performed system in SemEval-2010. It utilizes many external corpora to extract features from the sentence to build an SVM classifier.

| Model | Additional Features | F1 |
|---|---|---|
| SVM | POS, PropBank, morphological WordNet, TextRunner, FrameNet dependency parse, etc. | 82.2 |
| MV-RNN | POS, NER, WordNet | 81.8[2] |
| CNN | WordNet | 82.7 |
| FCM | NER | 83.0 |
| DT-RNN | NER | 73.1 |
| DepNN | WordNet | 83.0 |
| | NER | **83.6** |

Table 2: Results on SemEval-2010 dataset with Gigaword embeddings.

**MV-RNN** (Socher et al., 2012): This model finds the path between the two entities in the constituent parse tree and then learns the distributed representation of its highest node with a matrix for each word to make the compositions specific.

**CNN**: Zeng et al. (2014) build a convolutional model over the tokens of a sentence to learn the sentence level feature vector. It uses a special position vector that indicates the relative distances of current input word to two marked entities.

**FCM** (Yu et al., 2014): FCM decomposes the sentence into substructures and extracts features for each of them, forming substructure embeddings. These embeddings are combined by sum-pooling and input into a $softmax$ classifier.

**DT-RNN** (Socher et al., 2014) : This is an RNN for modeling dependency trees. It combines node's word embedding with its children through a linear combination but not a subtree embedding. We adapt the augmented dependency path into a dependency subtree and apply DT-RNN.

As shown in Table 2, DepNN achieves the best result (83.6) using NER features. WordNet features can also improve the performance of DepNN, but not as obvious as NER. Yu et al. (2014) had similar observations, since the larger number of WordNet tags may cause overfitting. SVM achieves a comparable result, though the quality of feature engineering highly relies on human experience and external NLP resources. MV-RNN models the constituent parse trees with a recursive procedure and its F1-score is about 1.8 percent lower than DepNN. Meanwhile, MVR-NN is very slow to train, since each word is associated with a matrix. Both CNN and FCM use features from the whole sentence and achieve similar performance. DT-RNN is the worst of all baselines, though it

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

[2] MV-RNN achieves a higher F1-score (82.7) on SENNA embeddings reported in the original paper.

also considers the information from shortest dependency paths and attached subtrees. As we analyze, shortest dependency paths and subtrees play different roles in relation classification. However, we can see that DT-RNN does not distinguish the modeling processes of shortest paths and subtrees. This phenomenon is also seen in a kernel-based method (Wang, 2008), where the tree kernel performs worse than the shortest path kernel. We also look into the DepNN model and find it can identify different patterns of words and the dependency relations. For example, in the *Instrument-Agency* relation, the word "using" and the dependency relation "prep_with" are found playing a major role.

## 5 Conclusion

In this paper, we propose to classify relations between entities by modeling the augmented dependency path in a neural network framework. We present a novel approach, DepNN, to taking advantages of both convolutional neural network and recursive neural network to model this structure. Experiment results demonstrate that DepNN achieves state-of-the-art performance.

## Acknowledgments

We thank all the anonymous reviewers for their insightful comments. This work was partially supported by National Key Basic Research Program of China (No. 2014CB340504), National Natural Science Foundation of China (No. 61273278 and 61370117), and National Social Science Fund of China (No: 12&ZD227). The correspondence author of this paper is Sujian Li.

## References

Nguyen Bach and Sameer Badaskar. 2007. A survey on relation extraction. *Language Technologies Institute, Carnegie Mellon University*.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A Shortest Path Dependency Kernel for Relation Extraction. In *North American Chapter of the Association for Computational Linguistics*.

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.

Aron Culotta and Jeffrey S. Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Meeting of the Association for Computational Linguistics*, pages 423–429.

Iris Hendrickx, Zornitsa Kozareva, Preslav Nakov, Sebastian Pad ok, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Meeting of the Association for Computational Linguistics*, pages 423–430.

Dat PT Nguyen, Yutaka Matsuo, and Mitsuru Ishizuka. 2007. Relation extraction from wikipedia using subtree mining. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1414. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259, Uppsala, Sweden, July. Association for Computational Linguistics.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5.

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, Jeju Island, Korea, July. 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.

Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 712–717. ACM.

Mengqiu Wang. 2008. A re-examination of dependency path kernels for relation extraction. In *IJCNLP*, pages 841–846.

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

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings* *of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.