A Richer-but-Smarter Shortest Dependency Path with Attentive Augmentation for Relation Extraction

Duy-Cat Can[1], **Hoang-Quynh Le**[1]*, **Quang-Thuy Ha**[1] **and Nigel Collier**[2]

[1]Faculty of Information Technology, VNU University of Engineering and Technology, Hanoi, Vietnam
[2]Department of Theoretical and Applied Linguistics, University of Cambridge, UK
{catcd,lhquynh,thuyhq}@vnu.edu.vn, nhc30@cam.ac.uk

Abstract

To extract the relationship between two entities in a sentence, two common approaches are (1) using their shortest dependency path (SDP) and (2) using an attention model to capture a context-based representation of the sentence. Each approach suffers from its own disadvantage of either missing or redundant information. In this work, we propose a novel model that combines the advantages of these two approaches. This is based on the basic information in the SDP enhanced with information selected by several attention mechanisms with kernel filters, namely RbSP (Richer-but-Smarter SDP). To exploit the representation behind the RbSP structure effectively, we develop a combined deep neural model with a LSTM network on word sequences and a CNN on RbSP. Experimental results on the SemEval-2010 dataset demonstrate improved performance over competitive baselines. The data and source code are available at https://github.com/catcd/RbSP.

1 Introduction

One of the most fundamental tasks in natural language processing, as well as in information extraction, is Relation Extraction (RE), i.e., determining the semantic relation between pairs of named entities or nominals in a sentence or a paragraph. Take the following sentences from the SemEval-2010 task 8 dataset (Hendrickx et al., 2009) as examples:

(i) We put the soured $[cream]_{e1}$ *in the butter* $[churn]_{e2}$ *and started stirring it.*

(ii) The agitating $[students]_{e1}$ *also put up a* $[barricade]_{e2}$ *on the Dhaka-Mymensingh highway.*

Here the nominals 'cream' and 'churn' in sentence (i) are of relation `Entity-Destination(e1,e2)` while nominals 'students' and 'barricade' in sentence (ii) are of relations `Product-Producer(e2,e1)`.

The research history of RE has witnessed the development as well as the competition of a variety of RE methodologies. All of them are proven to be effective and have different strengths by leveraging different types of linguistic knowledge, however, also suffer from their own limitations. Some early studies stated that the shortest dependency path (SDP) in dependency tree is usually concise and contains essential information for RE (Bunescu and Mooney, 2005; Fundel et al., 2006). By 2016, this approach became dominant with many studies demonstrating that using SDP brings better experimental results than previous approaches that used the whole sentence (Xu et al., 2015a,b; Mehryary et al., 2016; Cai et al., 2016; Le et al., 2018). However, using the SDP may lead to the omission of useful information (i.e., negation, adverbs, prepositions, etc.). Recognizing this disadvantage, some studies have sought to improve SDP approaches, such as adding the information from the sub-tree attached to each node in the SDP (Liu et al., 2015) or applying a graph convolution over pruned dependency trees (Zhang et al., 2018b).

Another approach to extract the relation between two entities is using whole sentence in which both are mentioned. This approach seems to be slightly weaker than using the SDP since not all words in a sentence contribute equally to classify relations and this leads to unexpected noises (Nguyen and Grishman, 2015). However, the emergence and development of attention mechanism (Bahdanau et al., 2015) has re-vitalized this approach. For RE, the attention mechanism is capable of picking out the relevant words concerning target entities/relations, and then we can find critical words which determine primary useful se-

*Corresponding author

Proceedings of NAACL-HLT 2019, pages 2902–2912
Minneapolis, Minnesota, June 2 - June 7, 2019.

mantic information (Zhou et al., 2016; Verga et al., 2018). We therefore need to determine the object of attention, i.e., nominals themselves, their entity types or relation label. However, conventional attention mechanism on sequence of words cannot make use of structural information on dependency tree. Moreover, it is hard for machines to learn the attention weights from a long sequence of input text.

In this work we propose an enhanced representation for relations that combines the advantages of the above approaches. Basically, we focus on condensed semantic and syntactic information on the SDP. Compensating for the limitations of the SDP may still lead to missing information so we enhance this with syntactic information from the full dependency parse tree. Our idea is based on fundamental notion that the syntactic structure of a sentence consists of binary asymmetrical relations between words (Nivre, 2005). Since these dependency relations hold between a head word (parent, predicate) and a dependent word (children, argument), we try to use all child nodes of a word in the dependency tree to augment its information. Depending on a specific set of relations, it will turn out that not all children are useful to enhance the parent node; we select relevant children by applying several attention mechanisms with kernel filters. This new representation of relation is named Richer-but-Smarter SDP (RbSP).

Recently, deep neural networks (DNNs) have been effectively used to learn robust syntactic and semantic representations behind complex structures. Thus, we propose a novel DNN framework which combines Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Convolutional Neural Networks (CNN) (LeCun et al., 1989) with a multi-attention layer.

Our work has three main contributions:

- We proposed a novel representation of relation based on attentive augmented SDP that overcomes the disadvantages of traditional SDP.
- We improved the attention mechanism with kernel filters to capture the features from context vectors.
- We proposed an advanced DNN architecture that utilizes the proposed Richer-but-Smarter Shortest Dependency Path (RbSP) and other types of linguistic and architectural features.

2 Related Work

RE has been widely studied in NLP community for many years. Unsupervised (Hasegawa et al., 2004; Yan et al., 2009; Quan et al., 2014), semi-supervised (Chen et al., 2006; Carlson et al., 2010; Ammar et al., 2017) and distant supervision (Verga et al., 2018; Ji et al., 2017) methods have been proven effective for the task of detecting relations from unstructured text. However, in this paper, we mainly focus on supervised approaches, which usually have higher accuracy. In earlier RE studies, researchers focused on extracting various kinds of linguistic features, including both syntactic features and semantic cues (Chan and Roth, 2010; Nguyen and Grishman, 2014). However, all the feature-based methods depend strongly on the quality of designed features from an explicit linguistic pre-processing step.

Based on the idea that SDPs contain the essential information for RE, many studies exploit it with several refinements. Typical refinements include negative sampling (Xu et al., 2015a) and BRCNN (Cai et al., 2016) which model the directed shortest path. Liu et al. (2015) suggested incorporating additional network architectures to further improve the performance of SDP-based methods, which uses a recursive neural network to model the sub-tree. Some works utilized information over the whole dependency tree, such as Li et al. (2017) used dynamic extended tree conditioned LSTM for RE and Panyam et al. (2018) exploited whole dependency graph for relation extraction in biomedical text.

Recently, with the introduction and development of attention mechanism, many works tend to use whole sentence or paragraph and focus on the most relevant information using attention technique. Some studies apply a single attention layer, that focus on the word itself (Shen and Huang, 2016; Zhang et al., 2018a); word position (Zhang et al., 2017) and global relation embedding (Su et al., 2018). Other works apply several attention layers, such as word, relation and pooling attention (Wang et al., 2016), multi-head attention (Verga et al., 2018) and word- and entity-based attention (Jat et al., 2017). Luo et al. (2018) used a bidirectional Long Short-Term Memory architecture with an attention layer and a tensor layer for organizing the context information and detecting the connections between two nominals.

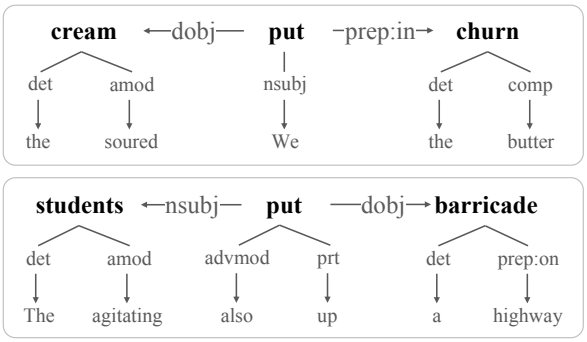

Figure 1: Examples of SDPs and attached child nodes.

3 Richer-but-Smarter SDP

As previously mentioned, we utilize the condensed information in the SDP to learn the relation between two nominals. The simple structure of the SDP is one of its weaknesses since there exists some useful information in dependency tree that does not appear in the SDP. This information can be leveraged to represent the relation more precisely. Two examples in Figure 1 belong to different relation types, but the paths between two nominals in these examples contain only one token ("**put**"). However, the meaning of token "**put**" in two SDPs are completely different. In this situation, it is difficult for the machine to distinguish the two shortest dependency paths from these instances.

We notice that the child nodes attached to the shortest dependency paths and their dependency relation from their parent can provide supplemental information for relation classification. In the previous examples, the sub-structure "$-$prt$\rightarrow$ **up**" provides semantic information about token '**put**" in the specific sentence to make it discriminated from the stand-alone one. Based on similar observations, we propose the idea of combining subtree information with original SDP to form a more precise structure for classifying relations. In this RbSP structure each token t is represented by itself and its attached children on the dependency tree.

4 Proposed Model

The overall architecture of our proposed model is shown in Figure 2. Given a sentence and its dependency tree, we build our model on the SDP between two nominals and its directed children on the tree. Here, we mainly focus on the SDP representation, which is composed of dependency embeddings, token embeddings, and token's augmented information. After SDP representation phase, each token and dependency relation is transformed into a vector. This sequence of vectors is then fed to a convolutional neural network to capture the convolved features that can be used to determine which relation two nominals are of.

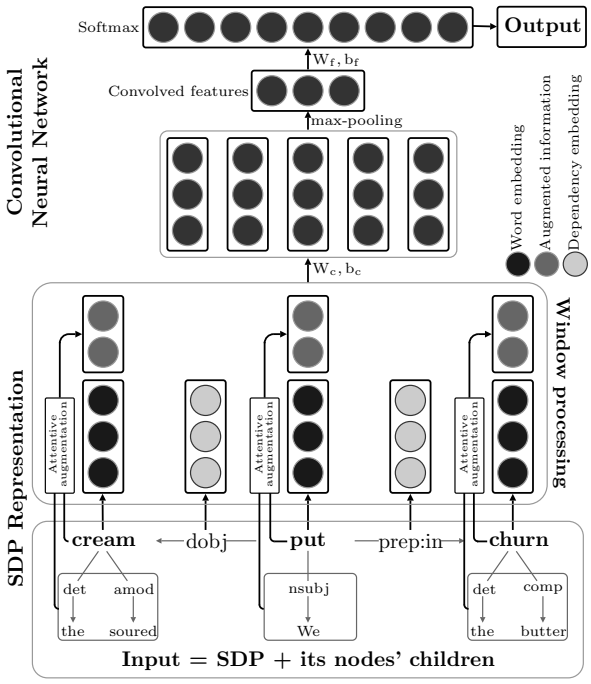

Figure 2: The architecture of RbSP model for relation classification. A CNN model is applied to the output of the SDP representation. Our proposed model takes the Augmented SDP between two nominals that includes dependencies, tokens and their children as input.

4.1 SDP Representation

The goal of this phase is to represent each component on the shortest path (dependency relation and token) by a corresponding vector. We concatenate the dependency type and dependency direction to form the embedding for a dependency relation, a non-linear transformer is followed to produce the final D-dimensional representation $\mathbf{d}_i \in \mathbb{R}^D$ of i-th dependency relation as follow:

$$\mathbf{d}_i = \tanh\left(\left[\mathbf{d}_i^{typ} \oplus \mathbf{d}_i^{dir}\right]\mathbf{W}_d + \mathbf{b}_d\right) \quad (1)$$

where $\mathbf{d}^{typ} \in \mathbb{R}^{dim_{typ}}$ and $\mathbf{d}^{dir} \in \mathbb{R}^{dim_{dir}}$ are dependency type and direction respectively; $\mathbf{W}_d \in \mathbb{R}^{(dim_{typ}+dim_{dir})\times D}$ and $\mathbf{b}_d \in \mathbb{R}^D$ are trainable parameters of the network.

For token representation, as mentioned above, we assume that each token should be interpreted by itself and its children. Then, the word information $\mathbf{t}_i$ of each token on the SDP is concatenated with its attentive augmented information $\mathbf{a}_i$ based on the attached children (which is calculated by Multi-layer attention with Kernel filters, see Section 4.2). In this work, we utilize four types of embeddings to represent the word information of each token, including:

- ***Pre-trained fastText embeddings*** (Bojanowski et al., 2017): which learned the word representation based on its external context.
- ***Character-based embeddings***: we use an internal LSTM to learn the information about word morphology (like the prefix or suffix).
- ***POS tag embeddings***: we embed the token's grammatical tag using a randomly initialized look-up table and update this parameter on model learning phase.
- ***WordNet embeddings***: which is in form of a sparse vector that figure out which basic WordNet synsets the token belongs to.

To take advantage of the original sentence sequence information, we use a recurrent neural network with LSTM units to pick up the information along the sentence $\mathbf{S} = \{\mathbf{t}_i\}_{i=1}^{n}$ as follow:

$$\mathbf{H} = \overleftrightarrow{\text{biLSTM}}(\mathbf{S}) = \{\mathbf{h}_i\}_{i=1}^{n} \quad (2)$$

Each token t_i is then augmented by the corresponding hidden state $\mathbf{h}_i$ from $\mathbf{H}$. Finally, this concatenation is transformed into an X-dimensional vector to form the representation $\mathbf{x}_i \in \mathbb{R}^X$ of the token. I.e.,

$$\mathbf{x}_i = \tanh\left(\left[\mathbf{t}_i \oplus \mathbf{a}_i \oplus \mathbf{h}_i\right]\mathbf{W}_x + \mathbf{b}_x\right) \quad (3)$$

where $\mathbf{W}_x$ and $\mathbf{b}_x$ are trainable parameters of the network.

4.2 Multi-layer attention with Kernel filters

To capture the appropriate augmented information from the child nodes of each token, we propose a novel multi-layer attention with kernel filters architecture. As illustrated in Figure 3, we employ two sequential attention layers on the children of a token to produce children context vectors. Afterward, to utilize all informative child nodes and preserve the integrity of the word information, we capture the token's augmented information using kernel filters instead of using the average of context vectors weighted by multi-layer attention.

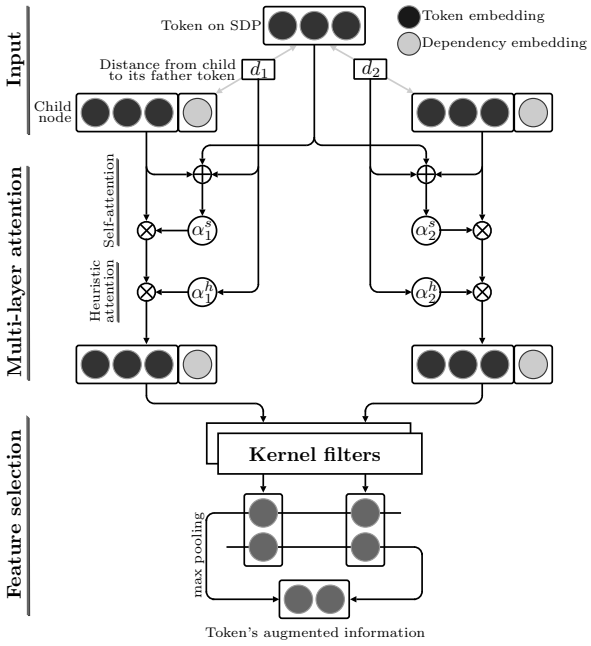

Figure 3: The multi-layer attention architecture to extract the augmented information from the children of a token on SDP. $\bigoplus$ denotes the concatenation of components. $\bigotimes$ denotes the scalar multiplication. Kernel filters are applied on context vectors to capture the features by convolution operation.

Given a token t and their child nodes, we first represent every token by a real-valued vector to provide lexical semantic features. Token t is transformed into a token embedding vector $\mathbf{t} \in \mathbb{R}^{dim}$ which is the concatenation of its word embedding and part-of-speech (POS) tag embedding. To utilize all the information in the sub-structure of token's children, we form a child node not only by its token embedding as in parent node but also by the dependency relation from its direct ancestor on the sentence's parse tree. Suppose t has a set C of M children, i.e., $C = \{c_1, c_2, ..., c_M\}$. Our model represents each child in C with a real-valued vector $\mathbf{c}_i \in \mathbb{R}^{dim+dim_{dep}}$. To additionally capture information about the child node to the target token, we incorporate the position embeddings d_i to reflect the relative distances between the i-th child's token to the target token on the original sentence.

We then apply a simple self-attentive network to child nodes $\{\mathbf{c_i}\}_{i=1}^{M}$ where the attention weights

are calculated based on the concatenation of themselves with parent information and distance from parent, as follow:

$$\begin{aligned}\bar{\mathbf{C}} &= \left\{\mathbf{c}_i \oplus \mathbf{t} \oplus d_i \mathbf{w}_d\right\}_{i=1}^{M} = \left\{\bar{\mathbf{c}}_i\right\}_{i=1}^{M} \\ \mathbf{e} &= \left\{\bar{\mathbf{c}}_i \mathbf{W}_e + b_e\right\}_{i=1}^{M} = \left\{e_i\right\}_{i=1}^{M} \\ \alpha_i^s &= \frac{\exp(e_i)}{\sum_{k=1}^{M} \exp(e_k)}\end{aligned} \tag{4}$$

where $\oplus$ denotes the concatenation operation; $\mathbf{w}_d \in \mathbb{R}^{dim_d}$ is the base distance embedding; $\mathbf{W}_e \in \mathbb{R}^{(2dim+dim_{dep}+dim_d)\times 1}$ and $b_e \in \mathbb{R}$ are weight and bias term. The self-attentive context vector $\mathbf{a}^s$ of the target token is the weighted sum of the self-attentive children context vectors based on the weights as follows:

$$\begin{aligned}\mathbf{c}_i^s &= \alpha_i^s \mathbf{c}_i \\ \mathbf{a}^s &= \sum_i \mathbf{c}_i^s\end{aligned} \tag{5}$$

We observe that the importance of a child node to the parent node depends on the distance between them on the original sentence. Therefore, we apply a heuristic attentive layer on the self-attentive children context vectors based on the distances $d_1, d_2, ..., d_M$ to keep track of how close each child is to the target token. We heuristically choose the activation function for the distances $d_1, d_2, ..., d_M$ as $f(d) = \beta d^2$ with $\beta = -0.03$, and a $\mathrm{softmax}$ layer is followed to calculate the heuristic attention weight. I.e.,

$$\begin{aligned}\alpha_i^h &= \frac{\exp(\beta d_i^2)}{\sum_{k=1}^{N} \exp(\beta d_k^2)} \\ \mathbf{c}_i^h &= \alpha_i^h \mathbf{c}_i \\ \mathbf{a}^h &= \sum_i \mathbf{c}_i^h\end{aligned} \tag{6}$$

The multi-attentive context vector $\mathbf{a}^h$ is a synthetic representation of all child nodes with the target token node taken into account. Since the child nodes are usually distinct from each other, an average vector is not suitable to represent the children information. We propose to use the kernel filters to capture the relevant and important information from the output of the multi-attention layer. K kernel filters are applied to each child's attentive vector to produce K features from each child. I.e.,

$$\mathbf{F} = \left\{\mathrm{ReLU}\left(\mathbf{c}_i^h \mathbf{W}_f + \mathbf{b}_f\right)\right\}_{i=1}^{M} \tag{7}$$

where $\mathbf{W}_f \in \mathbb{R}^{(2dim+dim_{dep}+dim_d)\times K}$ is the weight of K kernel filters; and $\mathbf{b}_f \in \mathbb{R}^K$ is bias term. Finally, to produce the final augmented information $\mathbf{a}$, we apply a max-pooling (Boureau et al., 2010) layer to the feature matrix $\mathbf{F}$ and select the most important features as follow:

$$\mathbf{a} = \left\{\max\left(\mathbf{F}_k^{\intercal}\right)\right\}_{k=1}^{K} \tag{8}$$

4.3 CNN on RbSP

After SDP representation layer, the input SDP is transformed into:

$$\mathbf{SDP} = \left[\mathbf{x}_1, \overleftarrow{\mathbf{d}_1}, \mathbf{x}_2, ..., \mathbf{x}_{N-1}, \overrightarrow{\mathbf{d}_{N-1}}, \mathbf{x}_N\right] \tag{9}$$

where the over arrow on $\mathbf{d}_i$ denotes the direction of the dependency relation. We build the CNN model on this $\mathbf{SDP}$; our model is similar to the model of Xu et al. (2015a). In general, let us define the vector $\mathbf{x}_{i:i+j}$ as the concatenation of j tokens and $j-1$ dependency relation between them. I.e.,

$$\mathbf{x}_{i:i+j} = \mathbf{x}_i \oplus \mathbf{d}_i \oplus \mathbf{x}_{i+1} \oplus ... \oplus \mathbf{d}_{i+j-2} \oplus \mathbf{x}_{i+j-1} \tag{10}$$

The convolution operation with region size r applies k filters to all possible window of r successive tokens to produce convolved feature map. We then gather the most important features by applying a max pooling (Boureau et al., 2010) layer over the entire feature map. I.e., the convolutional layer computes the i-th element of the convolved feature vector $\mathbf{f}$ as follows:

$$\mathbf{f}_i = \max_{0 \le j \le N-r+1} \left[\mathbf{x}_{j:j+r} \mathbf{W}_c + \mathbf{b}_c\right]_i \tag{11}$$

where $\mathbf{W}_c \in \mathbb{R}^{(rX+(r-1)D)\times k}$ and $\mathbf{b}_c \in \mathbb{R}^k$ are the weight matrix and bias vector of the convolutional layer. The output $\mathbf{f}$ of the convolutional layer is then fed to a softmax classifier to predict a $(K+1)$-class distribution over labels $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \mathrm{softmax}\left(\mathbf{f}\mathbf{W}_y + \mathbf{b}_y\right) \tag{12}$$

where $\mathbf{W}_y$ and $\mathbf{b}_y$ are parameter of the network to be learned.

4.4 Model Training

The proposed model can be stated as a parameter tuple $\theta = (\mathbf{W}, \mathbf{b})$. To compute the model parameters θ, we define the training objective for a data sample as:

$$L(\theta) = -\sum_{i=0}^{K} \mathbf{y}_i \log \hat{\mathbf{y}}_i + \lambda \|\theta\|^2 \qquad (13)$$

where $\mathbf{y} \in \{0, 1\}^{(K+1)}$ indicating the one-hot vector represented ground truth; and λ is a regularization coefficient. By minimizing $L(\theta)$ using mini-batch gradient descent (GD) with Adam optimizer (Kingma and Ba, 2014), θ is updated through neural network structures.

4.5 Additional techniques

For this paper, we directly utilize the pre-trained fastText word embeddings model (Bojanowski et al., 2017) which is trained on Wikipedia data. The look-up tables for dependency embeddings, word characters, POS tags are randomly constructed using the Glorot initializer (Glorot and Bengio, 2010) and are treated as the parameters to be learned during the training phase.

Since the CNN model takes the fixed size matrix as input, we pad the inputs in each batch of data dynamically to the longest input length of the batch. We further use the batch normalization (Ioffe and Szegedy, 2015) which is able to enable higher learning rates and reduces over-fitting.

During the training phase, we make use of several techniques, including: *clipping the gradients* if their norm exceeds a given threshold (Goldberg, 2017); applying *dropout* (Srivastava et al., 2014) with the probability of 0.5 on embeddings layer, CNN hidden states, and penultimate layer; and using *early stopping* (Caruana et al., 2001) by validation loss.

Further, to reduce the impact of random effects on our model, we employ the ensemble mechanism (Krogh and Sollich, 1997). For this study, we run the model for 20 times and uses the strict majority vote to obtain the final results.

5 Experimental Evaluation

5.1 Dataset

Our model was evaluated on SemEval-2010 Task 8 dataset (Hendrickx et al., 2009), which contains $10,717$ annotated relation classification examples and is separated into two subsets: $8,000$ instances for training and $2,717$ for testing. We randomly split 10 percents of the training data for validation. There are 9 directed relations and one undirected `Other` class.

We conduct the training-testing process 20 times and calculate the averaged results. For evaluation, the predicted labels were compared to the golden annotated data using standard precision (P), recall (R), and F1 score metrics.

5.2 Performance of the RbSP Model

Table 1 summarizes the performance of our model and comparative models. For a fair comparison with other researches, we implemented a baseline model, in which we remove all the proposed augmented information (multi-layer attention with kernel filters and LSTM on original sentence). This baseline model is similar to the model of Xu et al. (2015a) with some technical improvements and additional information sources. It yields higher F1 than competitors which are based on SDP without any data augmentation methods. This result is also comparative when is placed next to the result of basic Attention-CNN model.

The results also demonstrate the effectiveness of our proposed methods that brings an improvement of 1.5% in F1, compared to the baseline result. Our RbSP model yields an F1-score of 86.3%, outperforms other comparative models, except Multi-Att-CNN model of Wang et al. (2016) with multi-level attention CNN. However, we have tried to re-implement the Multi-Att-CNN, but we failed to reproduce the positive result in the original paper. The performance of our re-implementation is about 84.9% of F1. This result has a high consensus with Luo et al. (2018) since they also tried to re-build this model, and their re-implemented result is not much different from us, as 85.5%.

It is worth to note that when comparing with another augmented method of Liu et al. (2015), our multi-layer attention with kernel filters architecture brings more significant improvement. Relatively, in comparison of efficiency of augmented methods on the baseline model, the full-tree augmentation only brings 1% improvement of F1 while our attentive augmentation boosts up to 1.5%. Unlike the method of using the whole subtree to supplement information for the target node, our method only uses the most relevant nodes that are direct children to represent augmented infor-

Model	Source of information	F1
depLCNN (Xu et al., 2015a)	Word embeddings, SDP, CNN	81.9
	+ WordNet, word around nominals	83.7
	+ Negative sampling	85.6
BRCNN (Cai et al., 2016)	Word embeddings, SDP, LSTM, CNN	85.4
	+ POS, NER, WordNet embeddings, inverse SDP	86.3
DepNN (Liu et al., 2015)	200-d Gigaword embeddings, SDP, CNN	81.8
	+ Augmented sub-tree, Recursive Neural Network	82.8
	+ NER	83.6
Attention-CNN (Shen and Huang, 2016)	Sentence convolution, Attention-based context	84.3
	+ WordNet, Words around nominals	85.9
AT-BLSTM (Luo et al., 2018)	Word embeddings, Sentence attention features, Tensor feature	86.3
Multi-Att-CNN (Wang et al., 2016)	Multi-Level Attention CNNs, Attention pooling	88.0†
		85.5‡
Baseline	Word embeddings, POS tag, WordNet	84.8
RbSP (our model)	Baseline + Augmented Information	86.3
	+ ensemble	**86.7**

Table 1: The comparison of our RbSP model with other comparative models on SemEval-2010 task 8 dataset. The reported results are macro-averaged F1 scores of (9+1)-way evaluation with directionality taken into account. Since the comparative models did not report the precision (P) and recall (R), we also report the F1 score only. †: We failed to reproduce good result with the Multi-Att-CNN model, the performance of our implementation is just about 84.9. ‡: Another re-implemented result of Multi-Att-CNN model reported by Luo et al. (2018).

mation. In addition, our method further focuses on the most important children through two attention layers.

We also observe that during many training-testing processes, the results may vary. The standard deviation of 20 runs is about 0.27. We perform the ensemble strategy by majority voting on the results of 20 runs, and it drives our model to achieve a better result of 86.7%. This result is outperformed other comparative models.

5.3 Contribution of different components

Figure 4 shows the changes in F1 when removing each proposed component from the RbSP model. The F1 reductions illustrate the contributions of all proposals to the final result. However, the impact levels vary with different components. Between two proposed component, the multi-layer attention with kernel filters (augmented information) plays a vital role when contributing 1.22% to the final performance while the contribution of the LSTM on the original sentence is 0.33%.

An interesting observation comes from the interior of the multi-layer attention with kernel filters. The impact of removing the whole augmented information is much higher than the total impact of removing multi-layer attention or kernel filters (1.22 vs. $0.42+0.18 = 0.6$). These results demonstrate that the combination of constituent parts is thoroughly utilized by our sequential augmented architecture.

Another experiment is on investigating the meaning of each attention component. The result lightly reduces when we remove the self-attention or heuristic attention component. The results also prove that our proposed heuristic attention method is simple but effective. Its improvement is equivalent to the self-attention which is a complex attention mechanism. Among the input of multi-layer attention, the word embedding has a great influence on the model performance. However, children POS tag and relation to parent are also essential components to have the good results.

5.4 Results Analysis

We studied model outputs to analyze system errors in the cases of using the baseline model and using the proposed model with RbSP representation.

In Figure 5, we considered four types of errors: If the model makes a wrong decision and labels an `Other` relation (negative) as an actual relation (positive), it indicates 1 `FP` (False Positive) error.

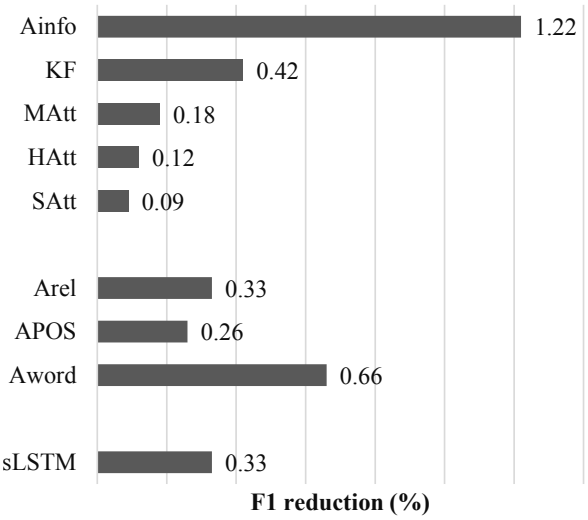

Figure 4: Comparing the contribution of proposed components by removing these components from the model: self-attention (SAtt), heuristic attention (HAtt), multi-layer attention (MAtt), kernel filers (KF), augmented information (Ainfo), augmentation using word embedding (Aword), augmentation using POS tag (APOS), augmentation using dependency relation (Arel), and LSTM on original sentence (sLSTM). F1 reduction is calculated by the average result of 20 runs.

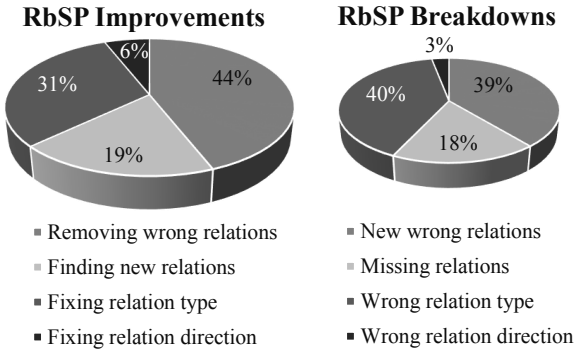

Figure 5: Comparing the effects of using RbSP in two aspects, (i) RbSP improved performance and (ii) RbSP yielded some additional wrong results. Four types of errors are analyzed, note that actual relations are considered as positive relations while `Other` is considered as negative: Labelling an `Other` relation as an actual relation; labelling an actual relation as `Other`; Confusion between types of relations; Direction errors.

Vice versa, if it labels an actual relation as `Other`, it brings 1 `FN` (False Negative). In the case that model confused between two types of relations, the model will be penalized twice, with 1 `FP` and 1 `FN`. Direction error, i.e., the model predicts the relation correctly but its direction wrongly, also brings 1 `FP` and 1 `FN`. The proportions of the left and the right of Figure 5 are quite consistent. In which, RbSP seems to have the most impact on determining whether an instance is positive or negative. RbSP also changes the decision of the relation type in quite many cases. It also influences the decision-making about relation's directionality, but not much.

Totally, the use of RbSP helps to correct more than 150 errors of the baseline model. However, it also yields some new errors (about 70 errors). Therefore, the difference of $F1$ between the baseline model and our RbSP model is only 1.5%, as stated in table 1.

Table 2 gives some realistic examples of different results when using the RbSP and not. We observed that the baseline model seems to be stuck in over-fitting problem, for examples, it classified all SDP with *prep:with* as `Instrument-Agency` and all SDP with *prep:in* as `Member-Collection` (examples $1-2$). RbSP is really useful for solving these cases partly since it uses attentive augmentation information to distinguish the same SDP or the same preposition with different meanings. RbSP is also proven to be stronger in examples $3-4$ to find new results and examples $5-7$ to fix wrong results. In our statistic, the use of RbSP bring the big advantage for the relations `Component-Whole`, `Message-Topic`, `Entity-Destination`, `Product-Producer` and `Instrument-Agency`. The results are almost constant for `Member-Collection` relations. Vice versa, we regret to state that using RbSb brings some worse results (examples $8-11$), especially for `Cause-Effect` and `Content-Container` relations.

Many errors seem attributable to the parser or our model's limitations that still cannot be overcome by using the RbSP (Examples $12-13$). We listed here some highlight problems to prioritize future researches (a) information on the SDP and its child nodes is still insufficient or redundant to make the correct prediction, (b) the direction of relations is still challenging since some errors appeared because we predict the relation correctly but its direction wrongly (c) the over-fitting problem (leading to wrong prediction - `FP`) and (d) lacking in generality (cannot predict new relation - `FN`).

#	SID†	SDP	Label*		
			Golden	RbSP	Baseline
1	8652	**Heating** *prep:with* **wood**	**Other**	**Other**	IA-21
2	10402	**officer** *prep:of* **college**	**Other**	**Other**	MC-12
3	9728	**news** *acl* **crashed** *nsubj* **plane**	**MT-12**	**MT-12**	Other
4	8421	**lane** *prep:on* **road**	**CW-12**	**CW-12**	Other
5	9092	**hurts** *prep:from* **memories**	**EO-12**	**EO-12**	CE-21
6	8081	**bar** *prep:of* **seats**	**CW-12**	**CW-12**	MC-21
7	10457	**show** *nsubj* **offers** *dobj* **discussion**	**MT-12**	**MT-12**	MT-21
8	10567	**stand** *prep:against* **violence**	**Other**	MT-12	**Other**
9	10296	**fear** *prep:from* **robbers**	**CE-21**	Other	**CE-21**
10	9496	**casket** *nsubjpass* **placed** *prep:inside* **casket**	**CC-12**	ED-12	**CC-12**
11	9734	**documents** *acl* **discussed** *prep:at* **meeting**	**MT-21**	MT-12	**MT-21**
12	9692	**rhyme** *prep:by* **thing**	**PP-12**	Other	Other
13	10562	**profits** *prep:from* **inflation**	**Other**	CE-21	CE-21

Table 2: The examples of error from RbSP and Baseline models. The predicted labels are from the best runs. †SIDs are sentence IDs in the testing dataset. *Abbreviation of relations: CC (Content-Container), CE (Cause-Effect), CW (Component-Whole), ED (Entity-Destination), EO (Entity-Origin), IA (Instrument-Agency), MC (Member-Collection), MT (Message-Topic), PP (Product-Producer). *Abbreviation of relation directions: 12 (e1,e2), 21 (e2,e1).

6 Conclusions

In this paper, we have presented RbSP, a novel representation of relation between two nominals in a sentence that overcomes the disadvantages of traditional SDP. Our RbSP is created by using multi-layer attention to choose relevant information to augment a token in SDP from its child nodes. We also improved the attention mechanisms with kernel filters to capture the features on the context vector. We evaluated our model on SemEval-2010 task 8 dataset, then compared the results with very recent state-of-the-art models. Experiments were also constructed to verify the rationality and effectiveness of each of the model's components and information sources. The results demonstrated the advantage and robustness of our model, includes the LSTM on the original sentence, combination of self-attention and heuristic mechanisms and several augmentation inputs as well. The analysis of the results still points our some weaknesses of the model. We aim to address them and further extensions of our model in future works. We released our source code and data on the public repository to support the re-producibility of our work and facilitate other related studies.

Acknowledgments

This research was supported by an EPSRC Experienced Researcher Fellowship (N. Collier: EP/M005089/1). We also thank the anonymous reviewers for their comments and suggestions.

References

Waleed Ammar, Matthew Peters, Chandra Bhagavatula, and Russell Power. 2017. The ai2 system at semeval-2017 task 10 (scienceie): semi-supervised end-to-end entity and relation extraction. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 592–596.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual

recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

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.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 756–765.

Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM.

Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 152–160. Association for Computational Linguistics.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2006. Relation extraction using label propagation based semi-supervised learning. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 129–136. Association for Computational Linguistics.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2006. Relexrelation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

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

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

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*, page 415. 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. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 448–456. JMLR. org.

Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2017. Improving distantly supervised relation extraction using word and entity based attention. In *6th Workshop on Automated Knowledge Base Construction (AKBC) at NIPS 2017*.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Anders Krogh and Peter Sollich. 1997. Statistical mechanics of ensemble learning. *Physical Review E*, 55(1):811.

Hoang-Quynh Le, Duy-Cat Can, Sinh T Vu, Thanh Hai Dang, Mohammad Taher Pilehvar, and Nigel Collier. 2018. Large-scale exploration of neural relation classification architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2266–2277.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Lishuang Li, Jieqiong Zheng, and Jia Wan. 2017. Dynamic extended tree conditioned lstm-based biomedical event extraction. *International Journal of Data Mining and Bioinformatics*, 17(3):266–278.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and WANG Houfeng. 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 (Volume 2: Short Papers)*, volume 2, pages 285–290.

Xiong Luo, Wenwen Zhou, Weiping Wang, Yueqin Zhu, and Jing Deng. 2018. Attention-based relation extraction with bidirectional gated recurrent unit and highway network in the analysis of geological data. *IEEE Access*, 6:5705–5715.

Farrokh Mehryary, Jari Björne, Sampo Pyysalo, Tapio Salakoski, and Filip Ginter. 2016. Deep learning with minimal training data: Turkunlp entry in the bionlp shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 73–81.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 68–74.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. *NAACL HLT 2015*, pages 39–48.

Joakim Nivre. 2005. Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32.

Nagesh C. Panyam, Karin Verspoor, Trevor Cohn, and Kotagiri Ramamohanarao. 2018. Exploiting graph kernels for high performance biomedical relation extraction. *Journal of biomedical semantics*, 9(1):7.

Changqin Quan, Meng Wang, and Fuji Ren. 2014. An unsupervised text mining method for relation extraction from biomedical literature. *PloS one*, 9(7):e102039.

Yatian Shen and Xuanjing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2526–2536.

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.

Yu Su, Honglei Liu, Semih Yavuz, Izzeddin Gur, Huan Sun, and Xifeng Yan. 2018. Global relation embedding for relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 820–830.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1298–1307.

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*, 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*, pages 1785–1794.

Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. 2009. Unsupervised relation extraction by mining wikipedia texts using information from the web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1021–1029. Association for Computational Linguistics.

Xiaobin Zhang, Fucai Chen, and Ruiyang Huang. 2018a. A combination of rnn and cnn for attention-based relation classification. *Procedia computer science*, 131:911–917.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018b. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.