Hierarchical graph attention network for relation extraction

Hongru Sun1*, Wancheng Ni1 and Yiping Yang1

1Institute of Automation, Chinese Academy of Sciences, Beijing, China

*E-mail: sunhongru2017@ia.ac.cn

Abstract. Previous research on relation extraction has proved the effectiveness of using dependency trees, which build non-local connections between tokens. However, the existing dependency-based models treat the dependency trees as an inherently flat graph, which causes a loss of dependency information for representing sentences. Besides, they fail to consider the fact that the importance of tokens on the dependency tree varies with different relations to be extracted. In this paper, we propose a novel hierarchical graph attention network HierGAT, which can generate multi-level dependency trees and extract key information from them to improve relation extraction. Specifically, it contains multiple dependency-based graph attention layers, each of which takes a different dependency tree generated by an adaptive subtree pruning strategy as input, and distinguishes the importance of different tokens in a dependency tree. Finally, HierGAT integrates the output token representations of each layer with a multi-head attention mechanism, and learns sentence representation for relation extraction through a pooling layer. The experimental results demonstrate that our method outperforms state-of-the-art baselines on the benchmark datasets.

1. Introduction

Relation extraction aims to recognize the correct relations between two entities from plain texts. It serves as a fundamental step in a variety of natural language processing applications, such as information extraction, knowledge base construction [1], question answering [2], etc. In previous research, much progress has been made on relation extraction, including traditional sequence-based methods [3, 4] and dependency-based[5, 6]. In contrast, dependency-based methods can achieve better performance, mainly because they can extract long-range syntactic relations that are obscure from the surface [7].

According to the scale of dependency trees incorporated, the dependency-based methods can be categorized into three classes: graph-level, substructure-level and token-level. Graph-level methods [8, 7] incorporate dependency trees as a graph into the models, whereas substructure-level [9, 10, 11] methods break up dependency trees into task-driven substructures, such as shortest paths and subtrees for relation extraction, and operate on these small structures. Token-level methods are mostly applied to feature engineering-based model [4], which feed dependency relation into classifiers as the feature of tokens. Obviously, the graph-level methods lose less structural information of dependency trees than the other two approaches, so it achieves better performance on relation extraction. However, existing graph-level methods suffer from several drawbacks, which can be divided into two aspects.

Firstly, they consider a dependency tree as a flat graph, ignoring some important structural information reflecting the dependency relations. A dependency tree contains subtrees of many different levels, for example in figure 1, the root node joined has four subtrees including Steve Ballmer,
the company, in 1980 and would later succeed Bill Gates as CEO. Furthermore, the token succeed is also a root node of four smaller subtrees, and we can iteratively find the subtree until it is indivisible. These subtrees can reflect the different-level structure information of dependency trees, which are complete semantic units with different importance for dependency-based relation extraction. However, existing graph-level methods only consider whether there exists a dependency relation between two tokens, but ignore the hierarchy of dependency trees, causing the learned representations of tokens to be insufficient.

Figure 1. An example of dependency tree for a sentence expressing a relation (CEO) between entities Steve Ballmer and the company. The shortest dependency path between these entities is highlighted in red. Each edge is marked with a dependency type, and the pruning distance of each token is shown in parenthesis.

Secondly, they fail to consider different importance of different tokens for extracting a specific relation. In graph-level models, the representation of a token is formed by the aggregation of the representations of connected tokens around it, also called neighborhood nodes. For a certain relation to be extracted, the importance of the neighborhood nodes for each token are obviously different. For example, in figure 1, the token succeed has four neighborhood nodes, including would, later, Bill and as, and the token as is more important than the other three tokens because it is relevant to the relation. However, existing graph-level methods consider neighborhood nodes to be equally important by applying graph convolutional networks (GCNs) [12], resulting in the key information of the task being smoothed.

Figure 2. The HierGAT Model with an Example Sentence.

As shown on the left, it is composed of \( L \) layers with the subtree pruning strategy and takes \( h^{(0)} \) as input, which contains pruning distance embeddings (green dots). In the HierGAT layer, the representation of each node is obtained by a weighted average and transform of its neighborhood nodes' representations, and the neighborhood weights are calculated based on self-attention and dependency types, as the example shown at bottom right. Top right shows an example of the subtree pruning, which takes the whole dependency tree as input and deletes the
lowest scored node along with its subtree. Eventually, a multi-head HierGAT layer is used to integrate a token’s multi-layer representations as one.

Based on the above analysis, we propose a novel hierarchical graph attention network (HierGAT), which is tailored for dependency-based relation extraction. It can capture the multi-level structure information of the dependency tree through multi-layered graph representation learning. Specifically, each HierGAT layer applies graph attention mechanism to calculate the different weights of neighborhood tokens and updates each token’s representation considering more critical information. Besides, the weights are also used to automatically screen out valuable subtrees between two HierGAT layers, called adaptive subtree-pruning, so that each layer can take different perspectives of the dependency tree as input. Then, it uses a multi-head attention to integrate a token’s multi-layer representations as one. This layer allows the final representation of each token to contain multi-level dependency information, which is more sufficient for relation extraction. Finally, we use masked max pooling to generate sentence representation for classification. To sum up, the main contributions of our work are summarized as follows:

1. We propose a novel end-to-end hierarchical graph attention network HierGAT, which captures different-level structural information of dependency trees for improving relation extraction.
2. Our model can learn different importance of different tokens in the dependency tree with an attention mechanism, which could also be used for adaptive subtree pruning to get multi-level dependency tree structures.
3. We conduct extensive experiments on two benchmark relation extraction datasets. The experimental results demonstrate the effectiveness of the proposed model over state-of-the-art methods.

2. Our method
In this section, we introduce our proposed hierarchical graph attention network model HierGAT for relation extraction, which distills multi-level structural information from the dependency tree through multiple graph representation layers. The task of relation extraction 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 types.

2.1. Hierarchical graph attention network
The HierGAT is composed of $L + 1$ graph representation layers as shown in the left of figure 2. Each layer is designed as a dependency-based graph attention mechanism to update the representation of tokens. Meanwhile, the deeper layer ($1 < l \leq L$) is equipped with an adaptive subtree pruning strategy, which is driven by the attention scores. The last multi-head HierGAT layer leverages the multi-head graph attention mechanism to integrate the former $L$ layers’ output, and generates the final representation of tokens. We will introduce each of these modules as follows.

2.1.1. One HierGAT layer
We start by describing a single HierGAT layer. Given an undirected graph of $N$ nodes $G = (\nu, \varepsilon)$ where $\nu$ and $\varepsilon$ are the set of nodes and edges, respectively. As shown in figure 2, the aggregation computation for node $i$ at the $l$-th layer, which takes all neighborhood nodes representations $\mathbf{h}^{(i)}_{\ell-1} \in \mathbb{R}^{F'}$ as input and outputs the representation $\mathbf{h}^{(i)}_{\ell} \in \mathbb{R}^{F'}$ ($F$ and $F'$ are hyper-parameters of the dimension), can be defined as:

$$\mathbf{h}^{(i)}_{\ell} = \sigma(\sum_{j \in \mathcal{N}^{(i)}(l)} \alpha^{(i)}_{iy} \mathbf{W}^{(i)} \mathbf{h}^{(i-1)}_{j} + \mathbf{b}^{(i)}_{y})$$

where $\mathcal{N}^{(i)}(l)$ is the set of one-hop neighborhood of node $i$ according to the dependency tree fed into the $l$-th layer, indicated as the adjacency matrix $\mathbf{A}^{(i)}$. $\alpha^{(i)}_{iy}$ is the neighborhood weight between nodes $i$ and $j$.  


\( W^{(i)} \in \mathbb{R}^{d \times r} \) is the weight matrix for linear transformation, \( b^{(i)} \) is the bias vector, and \( \sigma \) is an activation function (e.g., ELU [13]).

Before feeding the token representation into HierGAT, we first apply a bi-directional LSTM network to generate contextualized representations, and encode the shortest dependency path information for each token. Formally, the initial tokens representation \( h^{(0)} \in \mathbb{R}^{N \times d} \) is:

\[
h^{(0)} = Bi-LSTM(x) \parallel p
\]

where \( h^{(0)} \) and \( p \in \mathbb{R}^{N \times d} \) is the pruning distance embedding, which could be regarded as the feature of the shortest path for each token in the dependency tree.

The shortest path is considered to be the most useful dependency information for relation extraction [9]. It refers to the multi-hop path that connects \( e_i \) and \( e_z \) in the dependency tree. As shown in figure 1, Ballmer Steve \( \rightarrow \) joined \( \rightarrow \) the company is the shortest path. Previous efforts have attempted to extract the shortest path feature by the pruning distance \( pd \), which is defined as the distance (in terms of hop) between a token and the shortest path [7]. It can be inferred that the minimum value of \( pd \) is 0, which means the node is exactly on the shortest path, just like the three nodes Ballmer Steve, joined and the company in figure 1. Besides, the distance of the token off the dependency tree is set to positive infinity, such as the token and. However, these methods usually consider that the tokens with the large pruning distance are redundant. They only reserve the token within a default pruning distance and cut out the others in the dependency tree, resulting in the loss of useful information. For example, CEO is one of the farthest tokens from the shortest path in the dependency tree, but it is also the most critical token for relation extraction. In our model, we take full dependency trees as input and transfer the pruning distance into a kind of position embeddings of \( d_{pd} \) dimension for each token.

2.1.2. Graph attention mechanism
As shown in the bottom right of figure 2, the neighborhood weights are performed by the masked self-attention mechanism on the adjacent nodes. It should be noted that, GCN models ignore the neighborhood weights. The neighborhood weight at the \( l \)-th HierGAT layer \( \alpha^{(l)}_y \) is defined as:

\[
\alpha^{(l)}_y = \text{softmax}(e^{(l)}_y) = \frac{\exp(e^{(l)}_y)}{\sum_{i \in \mathbb{N}^{(l)}(i)} \exp(e^{(l)}_i)}
\]

where \( e^{(l)}_y \) denotes the attention score between nodes \( i \) and \( j \) at the \( l \)-th layer, and we only compute \( \alpha^{(l)}_y \) for nodes \( j \in \mathbb{N}^{(l)}(i) \). Considering the effects of dependency type, we define \( e^{(l)}_y \) as:

\[
e^{(l)}_y = a(W^{(l)}\tilde{R}_i^{(l-1)}, W^{(l)}\tilde{R}_j^{(l-1)}, \tilde{R}_y)
\]

where \( \tilde{R}_y \in \mathbb{R}^{d} \) is the embedding of the dependency relation type between nodes \( i \) and \( j \). \( a() \) is a shared attentional mechanism, which transforms \( \mathbb{R}^{(2^{2d})} \) to \( \mathbb{R} \).

2.1.3 Adaptive subtree pruning
The primary goal of adaptive subtree pruning is to enable the HierGAT model to select relevant subtrees for each layer, as shown in the bottom right of figure 2. It automatically reserves the top \( K \) highest score nodes and removes the rest of nodes along with their subtrees based on the attention scores. Eventually, each HierGAT layer contains different subtrees, which can be regarded as a multi-level description of the dependency tree structure. The adaptive subtree pruning operation consists of two steps: subtree rating and subtree selection, which will be elaborated as follows:

**Subtree rating** is designed to measure the importance of the root node of each subtree (A leaf node of the dependency tree are also regarded as a subtree). We utilize two outputs of each HierGAT layer,
attention scores matrix and the representation matrix of all nodes, to calculate the subtree rating in the l-th layer \( r^{(l)} \in \mathbb{R}^k \), which is defined as:

\[
 r^{(l)} = \rho(\mathbf{E}^{(l-1)i} \mathbf{h}^{(l-1)i} \mathbf{W}_e^{(l-1)})
\]

where \( \rho \) denotes the activation function (e.g. tanh), \( \mathbf{E}^{(l-1)i} \in \mathbb{R}^{n \times r} \) is the attention score matrix of the \((l-1)\)-th layer’s output and each of its elements is calculated by Equation , \( \mathbf{h}^{(l-1)i} \in \mathbb{R}^{n \times r} \) is the node representation matrix of the \((l-1)\)-th layer's output. \( \mathbf{W}_e^{(l-1)} \in \mathbb{R}^{r \times d} \) is the only parameter of the adaptive subtree pruning.

**Subtree selection** aims to delete the low-value nodes along with their subtrees based on the \( r^{(l)} \) for each layer. Since the length of each sentence is different, we design a pruning ratio \( \eta \in (0,1] \) to determine the number of root nodes to be kept \( K \) at \( l \)-th layer:

\[
 K^{(l)} = \left\lceil \sum M^{(l-1)} \times \eta \right\rceil
\]

where \( M^{(l-1)} \) denotes the number of node input to the \((l-1)\)-th layer. Then the \( K \) nodes are selected based on the value of \( r^{(l)} \):

\[
 \text{idx}^{(l)} = \text{top-rank}(r^{(l)}, K^{(l)})
\]

where \text{top-rank} is the function that returns the indices of nodes, whose values belong to the top \( K \) at the \( l \)-th layer. Finally, we apply a mask to generate a new adjacency matrix \( \mathbf{A}^{(l)} \):

\[
 \mathbf{A}^{(l)} = \begin{cases} 
   \mathbf{A}^{(l-1)} & \text{if } i \in \text{idx}^{(l)} \\
   0 & \text{else}
\end{cases}
\]

It is worth stressing that, if a node is pruned at \( l \)-th layer, its representation at \( l \)-th layer will be 0 because all of its neighborhood nodes have also been deleted according to Equation.

### 2.1.4 Multi-head HierGAT layer
To integrate the former \( L \) layers’ output, the multi-head HierGAT layer is defined as:

\[
 \hat{h}_i = \left\| \sigma \left( \sum_{j \in N(i)} \mathbf{a}_j \mathbf{W} \hat{h}_j + \hat{b} \right) \right\|
\]

where \( \| \) denotes the vector concatenation operation and \( \hat{h}_i \in \mathbb{R}^r \) denotes the final representation of the node \( i \). \( N(i) \) is the set of one-hop neighborhood nodes of node \( i \) according to the initial dependency tree. Note that, \( \mathbf{W} \) consists of \( L \times F \) features (rather than \( F \)) for each token.

### 2.2. Relation extraction
After demonstrating the HierGAT model over the dependency tree, we obtain hidden representations of all tokens. Given these representations, the goal of relation extraction is to predict the relation between two entities \( e_1 \) and \( e_2 \). We concatenate the sentence representation and entity representations to get the final representation for classification. First, we obtain the sentence representation \( \hat{h}_\text{seni} \) as follows:

\[
 \hat{h}_\text{seni} = f(\mathbf{h}) = f(\text{HierGAT}(\mathbf{x}))
\]

where \( f : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^d \) is a masked max pooling function that ignores the entities vectors and maps from the other tokens’ vectors to 1 sentence vector. Then, we also obtain the representations of entity \( e_1 \) and \( e_2 \), which are marked as \( \hat{h}_{e_1} \) and \( \hat{h}_{e_2} \) by masked max pooling separately. The entity representations will be concatenated with sentence representation to form a new representation. Finally, we apply a feed-forward neural network (FFNN) over the concatenated representations inspired by relational reasoning works [14]:

\[
 \hat{h}_\text{final} = \text{FFNN}( [\hat{h}_\text{seni}, \hat{h}_{e_1}, \hat{h}_{e_2} ])
\]
where $\hat{h}_{\text{final}}$ is taken as input to a softmax regression classifier to predict the relation type.

3. Experiments

In this section, we first describe the experimental relation extraction datasets, baseline models and experimental settings. Then, the experiments of our proposed model and baseline models are carried out on the two datasets. Finally, we compare the performance of \textit{HierGAT} with different components and explain the effectiveness of our model by visualizing the attention mechanism.

3.1. Dataset

We conduct the experiments on the following two benchmark relation extraction datasets:

(1) \textbf{TACRED} [4] contains over 106k mention pairs drawn from the yearly \textit{TAC KBP4} challenge. It introduces 41 relation types and a special "no relation" type when the mention pair does not have a relation between them within these categories. Subject mentions are categorized into person and organization, and object mentions are categorized 16 fine-grained types (e.g., date and location). We report micro-averaged F1 scores on this dataset as following previous works [4].

(2) \textbf{SemEval-2010 Task 8} [15] focuses on semantic relations between pairs of nominals. For example, cancers and exposures are in an \textit{Cause-Effect} relation in "\textit{those cancers were caused by radiation exposures}". \textit{SemEval} covers 9 relation types and instances don’t fall in any of these types are labeled as \textit{Other}. The official evaluation metric is the macro-averaged F1-score (excluding \textit{Other}).

3.2. Baseline models

To validate our method, we compare it with several competitive dependency-based and neural sequence models.

**Dependency-based models** are divided into three categories: (1) A graph-level method \textit{C-GCN} [7], which treats the dependency tree as a graph and applies \textit{GCN} to extract structure information. \textit{C-AGGCN} [8], which improves \textit{C-GCN} by using attention mechanism to remove useless tokens. (2) Substructure-level methods, such as \textit{SDP-LSTM} [16], \textit{Tree-LSTM} [17], \textit{SPTree} [18] and \textit{DeepNN} [10], which break up the dependency tree into the shortest path and subtrees, and utilize \textit{RNN} or \textit{CNN} to capture sequence feature for relation extraction. (3) Token-level methods including \textit{LR} [4] and \textit{SVM} [19], which regard the dependency information as a kind of token's features.

**Sequence-based model.** We choose the state-of-the-art Position Aware LSTM (\textit{PA-LSTM}) [4], which outperforms several CNN and dependency-based models, as a strong baseline.

3.3. Experimental setup

In the data preprocessing step, we use the Stanford \textit{Parser} [20] to generate dependency trees and other token features (e.g., pos parsing), and leverage 300-dimensional \textit{GloVe} [21] vectors as the initialization for word embeddings $x$. We tune the following hyper-parameters according to results on the development sets. We set the embedding dimensions for both pruning distance $d_{\text{pruning}}$ and dependency type $d_t$, to 50, and set the hidden dimension ($F=F^\prime$) to 200. In the subtree pruning operation, we set the pruning ratio $\eta$ to 0.7, which means keeping 70% of the nodes according to the previous layer. In the output layer, we use max-pooling to generate sentence representation and a 2-layer \textit{FFNN} to predict the relation type. Experiments on the development sets show that the number of \textit{HierGAT} layers $L$=3 gives the best results on both datasets. For optimization, parameters are trained using the \textit{SGD} optimizer with an initial learning rate of 0.5 and a decay factor (0.9 for \textit{TACRED} and 0.95 for \textit{SemEval-2010}).

3.4. Results and analysis

The results on the \textit{TACRED} and \textit{SemEval-2010} datasets are shown in table 1 and table 2, respectively. Compared to dependency-based models, we can find that graph-level models, including \textit{HierGAT}, \textit{C-AGGCN} and \textit{C-GCN}, obtain higher F1 scores. We believe the reason behind this is that, these models capture the structure of the whole dependency tree by means of graph neural network, leading to less
dependency information loss. At the same time, HierGAT outperforms significantly better than both C\text{-}GCN and C\text{-}AGGCN, and achieves a new state of the art.

Table 1. Results on TACRED dataset.

| Method            | Precision | Recall | F1     |
|-------------------|-----------|--------|--------|
| C-GCN [7]         | 70.9%     | 63.0%  | 66.7%  |
| C-AGGCN [8]       | 73.6%     | 63.1%  | 67.9%  |
| SDP-LSTM [16]     | 66.3%     | 52.7%  | 58.7%  |
| LR [4]            | 73.5%     | 49.9%  | 59.4%  |
| Tree-LSTM [17]    | 66.0%     | 59.2%  | 62.4%  |
| PA-LSTM [4]       | 65.7%     | 64.5%  | 65.1%  |
| HierGAT(ours)     | 73.8%     | 64.7%  | 68.9%  |

Table 2. Results on SemEval-2010 dataset.

| Method            | F1     |
|-------------------|--------|
| C-GCN [7]         | 84.5%  |
| C-AGGCN [8]       | 84.7%  |
| SDP-LSTM [16]     | 83.7%  |
| DeepNN [10]       | 83.6%  |
| SVM [19]          | 82.2%  |
| SPTree [18]       | 84.4%  |
| PA-LSTM [4]       | 82.7%  |
| HierGAT(ours)     | 85.6%  |

Table 3. Ablation study for HierGAT model on TACRED.

| Model       | F1     | F1(w/o dependency type) | F1(w/o shortest path) | F1(w/o subtree pruning) |
|-------------|--------|-------------------------|-----------------------|-------------------------|
| HierGAT-1layer | 66.8%  | 66.5%                   | 65.3%                 | 66.8%                   |
| HierGAT-2layer | 67.5%  | 67.3%                   | 66.5%                 | 67.3%                   |
| HierGAT-3layer | 68.9%  | 68.4%                   | 67.0%                 | 66.7%                   |
| HierGAT-4layer | 68.2%  | 67.6%                   | 67.0%                 | 66.2%                   |

There are three reasons that contribute to the results: (1) HierGAT layer makes good use of more dependency information than C\text{-}GCN and C\text{-}AGGCN, because it takes into account the neighborhood weights, which are measured based on the dependency type and the pruning distance. (2) The subtree-pruning characterizes the dependency tree from whole to part, making HierGAT capture the multi-level structural information of the dependency tree comprehensively and concretely. (3) HierGAT integrates multi-level structural information together and generates a hierarchical representation for each token.

3.5. Study of our proposed model

To study the contribution of each component in the HierGAT, we first run an ablation study on the TACRED testset.

In table 3, we range the number of HierGAT layers from 1 to 4 and explored the performance of variants of our model removing dependency types, shortest paths and subtree pruning separately. It should be noted that, Multi-head HierGAT layer is also counted in the total number of layers. We summarize the analysis into four aspects as follows:

(1) With the increase of the number of layers, the performance of HierGAT improves until the number of layers is 3. In general, the single-layer HierGAT model can achieve a strong performance. As the number of layers increases, the ability to capture global information [22], [23] raises. Therefore, it can be proved that our model can capture the relevant structure information of dependency trees very well.
(2) Dependency type is useful for relation extraction, but its contribution is relatively small. It's probably because the dependency type is not a very effective task-driven feature, which has been discussed in DeepNN [10].

(3) Shortest dependency path is an efficient task-driven feature for relation extraction. We treat this feature as position encoding rather than rule-based pruning, allowing HierGAT to avoid deleting trigger token by mistake and reserve more valid information.

![Figure 3. The 1st HierGAT layer’s attention heat map.](image)

![Figure 4. The 2nd HierGAT layer’s attention heat map.](image)

(4) Subtree-pruning has the greatest impact on performance, furthermore, the impact increases as HierGAT deepens. This comparison can prove the effectiveness of subtree-pruning strategy. Besides, HierGAT-1 layer achieves the state-of-the-art result, showing that a single HierGAT layer is very efficient in extracting relevant information.

3.6. Case study
We conduct a case study of each HierGAT layer’s (except for the last Multi-head HierGAT layer) to show the visualization of two layer’s attention heat maps in figure 3 and figure 4, and explain why our model works. In the sentence ""There is a sense of pride in who he is." said Marc Morial, president of the National Urban League.", e₁ is the token National Urban League (in red), e₂ is the token Marc Morial (in blue) and the relation to be extracted is "org:top_members/employees".

The heat map shows the neighborhood weights of each token in rows, for example, the token "president" has two neighborhood nodes, "Morial" and "League", with different weights in both two HierGAT layers. We summarize two key findings: (1) "There is a sense of pride in who he is." has very little effect on relation extracting. (2) "president" is the most important token for extracting the relation "org:top_members/employees".

Comparing figure 4 with figure 3, on one hand, some root nodes of subtrees, including "There", "sense", "in", "who", "he", "is" and ".", have been eliminated by the adaptive subtree pruning layer. Therefore, representations of these low-value root nodes are not updated in the 2nd and subsequent HierGAT layers, meanwhile, the updates of other nodes are not disturbed. In this way, each layer reflects different subtrees of the dependency tree and generates hierarchical representation of each token. On the other hand, the weights of "president" among the neighborhood nodes of "Morial" and "League" have increased significantly in the 2nd HierGAT layer. So the representations of "Morial" and "League" will aggregate more semantics of "president", and we can infer from Equation that HierGAT should focus more on "president" to predict the relation type.

4. Related work
In this section, we will briefly describe the previous research related to our work in the field of relation extraction and graph convolutional networks.

**Relation Extraction.** Early efforts are based on feature engineering methods. Feature engineering methods extract various kinds of linguistic features and employ them in a statistical classifier like Max Entropy [24] and SVM [25]. Then, kernel-based methods such as tree-based kernels [26] and dependency path-based kernels [9] are explored to extract the relation. They provide an important insight that the shortest path between the two entities concentrates most of the information for identifying the relation between them. Dependency subtrees [27] are used in relation extraction by
modeling the subtrees between entities and keywords. Besides, sequence-based models, which apply convolutional neural networks (CNN) [28, 3] or recurrent neural networks (RNN) [4, 6] only on token sequence for relation extraction, achieve better performance than previous works.

Recently, dependency-based approaches, which try to incorporate structural information into neural networks, have attracted many researchers. DeepNN [10] combines the shortest dependency path as well as the dependency subtree into an augmented dependency path (ADP) structure, and uses CNNs and RNNs to encode them separately. Similarly, SDP-LSTM [16] leverages long short term memory networks (LSTM) to model shortest dependency paths for relation extraction. In addition, the Tree-LSTM [17] and SPTree [18] apply LSTM over the lowest common ancestor (LCA) subtree of two entities. The above four models can be considered as extracting semantic features at substructure-level of dependency trees. At present, the state-of-the-art method C-AGGCN [8], which is a modified version of C-GCN [7], incorporates full dependency trees into the model and utilizes attention mechanism to prune them. However, this method fails to capture keywords and multi-level structural information of dependency trees.

**Graph Convolutional Networks.** GCNs [12] play a significant role in representing the nodes in arbitrary structured graphs. It transforms the convolution into the operation on a first-order neighborhood around each node based on the spectral graph theory. But it is constrained by the inability to measure the importance of neighborhood nodes. More recently, graph attention networks (GATs) [29] were proposed to summarize neighborhood states by using masked self-attention layers [30]. Because of the representing smoothness, these two models cannot have too many layers, which prevents them from passing information into higher-order neighborhood nodes [23]. The motivation for our work is to obtain comprehensive multi-level information of dependency tree for optimizing the representation of the sentence.

5. Conclusions

In this paper, we propose a novel multi-layer hierarchical graph attention network HierGAT, which can extract multi-level structural dependency information from the dependency tree of a sentence. Each layer of our model updates the representation of each token based on its weighted neighborhood nodes. The weight is calculated by graph attention mechanism, which encodes dependency types and shortest paths and performs self-attention on the tokens. Additionally, we design an adaptive subtree pruning strategy, which can help each layer weed out uninformative subtrees automatically. To enhance HierGAT's multi-level perception of the dependency tree's structure, we leverage the multi-head attention mechanism to incorporate all graph representation layers. Finally, it learns sentence representation for relation extraction through a pooling layer. We conduct sound experiments on two benchmark relation extraction datasets. The experimental results demonstrate that our model outperforms other state-of-art baselines on both datasets.

References

[1] Chen, J., Tandon, N., & De Melo, G. (2015, December). Neural word representations from large-scale commonsense knowledge. In 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT) (Vol. 1, pp. 225-228). IEEE.

[2] Yu, M., Yin, W., Hasan, K. S., Santos, C. D., Xiang, B., & Zhou, B. (2017). Improved neural relation detection for knowledge base question answering. arXiv preprint arXiv:1704.06194.

[3] Zeng, D., Liu, K., Lai, S., Zhou, G., & Zhao, J. (2014). Relation classification via convolutional deep neural network.

[4] Zhang, Y., Zhong, V., Chen, D., Angeli, G., & Manning, C. D. (2017, September). Position-aware attention and supervised data improve slot filling. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 35-45).

[5] Bach, N., and Badaskar, S. (2007). A review of relation extraction. Literature review for Language and Statistics II. 2, 1-15.

[6] Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., & Xu, B. (2016, August). 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) (pp. 207-212).

[7] Zhang, Y., Qi, P., & Manning, C. D. (2018). Graph convolution over pruned dependency trees improves relation extraction. arXiv preprint arXiv:1809.10185.

[8] Guo, Z., Zhang, Y., & Lu, W. (2019). Attention guided graph convolutional networks for relation extraction. arXiv preprint arXiv:1906.07510.

[9] Bunescu, R. C., & Mooney, R. J. (2005, October). A shortest path dependency kernel for relation extraction. In Proceedings of the conference on human language technology and empirical methods in natural language processing (pp. 724-731). Association for Computational Linguistics.

[10] Liu, Y., Wei, F., Li, S., Ji, H., Zhou, M., & Wang, H. (2015). A dependency-based neural network for relation classification. arXiv preprint arXiv:1507.04646.

[11] Xu, K., Feng, Y., Huang, S., & Zhao, D. (2015). Semantic relation classification via convolutional neural networks with simple negative sampling. arXiv preprint arXiv:1506.07650.

[12] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

[13] Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289.

[14] Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., & Lillicrap, T. (2017). A simple neural network module for relational reasoning. In Advances in neural information processing systems (pp. 4967-4976).

[15] Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Séaghdha, D. O., Padó, S., et al. (2010, July). Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In Proceedings of the 5th International Workshop on Semantic Evaluation (pp. 33-38). Association for Computational Linguistics.

[16] Xu, Y., Mou, L., Li, G., Chen, Y., Peng, H., & Jin, Z. (2015, September). Classifying relations via long short term memory networks alone shortest dependency paths. In Proceedings of the 2015 conference on empirical methods in natural language processing (pp. 1785-1794).

[17] Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075.

[18] Miwa, M., & Bansal, M. (2016). End-to-end relation extraction using lstms on sequences and tree structures. arXiv preprint arXiv:1601.00770.

[19] Rink, B., & Harabagiu, S. (2010, July). Utd: Classifying semantic relations by combining lexical and semantic resources. In Proceedings of the 5th International Workshop on Semantic Evaluation (pp. 256-259). Association for Computational Linguistics.

[20] Klein, D., & Manning, C. D. (2003). Fast exact inference with a factored model for natural language parsing. In Advances in neural information processing systems (pp. 3-10).

[21] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[22] Hu, F., Zhu, Y., Wu, S., Wang, L., & Tan, T. (2019). Semi-supervised Node Classification via Hierarchical Graph Convolutional Networks. arXiv preprint arXiv:1902.06667.

[23] Li, Q., Han, Z., & Wu, X. M. (2018, April). Deeper insights into graph convolutional networks for semi-supervised learning. In Thirty-Second AAAI Conference on Artificial Intelligence.

[24] Kambhatla, N. (2004, July). 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 (pp. 22-es).

[25] GuoDong, Z., Jian, S., Jie, Z., & Min, Z. (2005, June). Exploring various knowledge in relation extraction. In Proceedings of the 43rd annual meeting on association for computational
linguistics (pp. 427-434). Association for Computational Linguistics.

[26] Culotta, A., & Sorensen, J. (2004, July). Dependency tree kernels for relation extraction. In Proceedings of the 42nd annual meeting on association for computational linguistics (p. 423). Association for Computational Linguistics.

[27] Nguyen, D. P., Matsuo, Y., & Ishizuka, M. (2007, July). Relation extraction from wikipedia using subtree mining. In Proceedings of the National Conference on Artificial Intelligence (Vol. 22, No. 2, p. 1414). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[28] Wang, L., Cao, Z., De Melo, G., & Liu, Z. (2016, August). Relation classification via multi-level attention cnns. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 1298-1307).

[29] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

[30] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).