Relation Extraction with Convolutional Network over Learnable Syntax-Transport Graph

Kai Sun,[1,2] **Richong Zhang,**[1,2*] **Yongyi Mao,**[3] **Samuel Mensah,**[1,2] **Xudong Liu**[1,2]
[1]SKLSDE, School of Computer Science and Engineering, Beihang University, Beijing, China
[2]Beijing Advanced Institution on Big Data and Brain Computing, Beihang University, Beijing, China
[3]School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada
sunkai@buaa.edu.cn, zhangrc@act.buaa.edu.cn, ymao@uottawa.ca, samensah@buaa.edu.cn, liuxd@act.buaa.edu.cn

Abstract

A large majority of approaches have been proposed to leverage the dependency tree in the relation classification task. Recent works have focused on pruning irrelevant information from the dependency tree. The state-of-the-art Attention Guided Graph Convolutional Networks (AGGCNs) transforms the dependency tree into a weighted-graph to distinguish the relevance of nodes and edges for relation classification. However, in their approach, the graph is fully connected, which destroys the structure information of the original dependency tree. How to effectively make use of relevant information while ignoring irrelevant information from the dependency trees remains a challenge in the relation classification task. In this work, we learn to transform the dependency tree into a weighted graph by considering the syntax dependencies of the connected nodes and persisting the structure of the original dependency tree. We refer to this graph as a syntax-transport graph. We further propose a learnable syntax-transport attention graph convolutional network (LST-AGCN) which operates on the syntax-transport graph directly to distill the final representation which is sufficient for classification. Experiments on Semeval-2010 Task 8 and Tacred show our approach outperforms previous methods.

Introduction

Relation classification aims to classify the semantic relations between two entities in a sentence. For instance, given a sentence *"My apartment has a pretty large kitchen."* and two entities *apartment* and *kitchen*, the goal is to classify the relation of the two entities into a *Component-Whole* category, denoted as *Component-Whole(kitchen, apartment)* . Relation classification plays a vital role in a variety of downstream natural language processing task including knowledge extraction from unstructured texts and question answering.

With the aim to address the classification task, several methods have been developed. The sequence-based methods only take word sequence as input (Zeng et al. 2014; Zhang et al. 2015) and attention mechanism is applied to distill relevant information for classification (Zhou et al. 2016; Shen and Huang 2016; Zhang et al. 2017; Lee, Seo, and Choi 2019). However, conventional attention mechanism on sequence of words ignores the grammar structure information of the sentence.

(a) *The **man** used a **support pillow** to remedy the **pain** caused by **stresses**.*
(b) *Amateur video shows some of the **devastation** caused by the tsunami **waves**.*
(c) *Work was completed by **contractors** using a temporary **entrance** off old Renwick road.*
(d) ***Analysts** assess distribution and changes in distribution over time by using **frequency**.*

Table 1: Four sentences with entities in bold.

So far, a variety of methods have been proposed to leverage dependency grammar information in this task (Miwa and Bansal 2016; Yang et al. 2016; Zhang, Qi, and Manning 2018; Guo, Zhang, and Lu 2019). In particular, the shortest dependency path (SDP) between the two entities in the dependency tree of the given sentence is believed to convey informative clues of target relation (Xu et al. 2015). However, some significant nodes might not be included in the SDP and be omitted (Liu et al. 2015). For example, the preposition is commonly excluded from SDP while it generally indicates the subject-predicate relationship between the considered entities.

Recent works have focused on pruning irrelevant information from the dependency tree. The C-GCN (Zhang, Qi, and Manning 2018) obtain a pruned tree by including tokens that are up to distance K away from the dependency path in the lowest common ancestor (LCA) subtree. However, the distance K is predefined and fixed. Such a rule-based pruning strategy is not general for most cases.

The state-of-the-art C-AGGCN (Guo, Zhang, and Lu 2019) exploits self-attention to transform the dependency tree into a fully connected weighted graph. However, as all the nodes are connected, the new introduced connections damage the dependency relationship between words preserved in the tree structure. How to effectively make use of relevant information while ignoring irrelevant information from the dependency trees remains a challenge in this task.

*Corresponding author:zhangrc@act.buaa.edu.cn

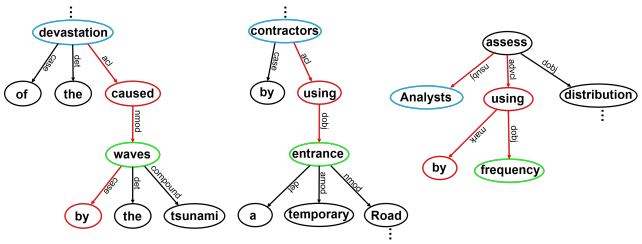

Figure 1: The substructures of three dependency trees. The entity relations of these three instances are *Cause-Effect(e2,e1)*, *Instrument-Agency(e2,e1)* and *Instrument-Agency(e2,e1)*. The two entities *e1* and *e2* are marked in blue and green separately. The most relevant nodes and edges for the relation classification task are marked in red.

Generally, we claim that it is necessary to model the dependency structure explicitly. One reason is that there may exist multiple entity pairs in one sentence. For example, consider the sentence (a) in Table 1. The target relations include *Instrument-Agency(support pillow,man)* and *Cause-Effect(stresses,pain)*. If the dependency structure is not explicitly modeled, the representation acquired by merely modeling the word sequence of sentences may be a mixture containing the feature of both relations. This mixed representation may provide confusing clues for relation classification.

Besides, the same grammar structure has relatively different importance in a different context. Consider the sentence (b) and sentence (c) shown in Table 1, their corresponding dependency trees are shown in Figure 1. The word node *by* and dependency edge *case* indicate the subject-predicate relation between two entities *waves* and *devastation*, which is significant for the prediction of the *Cause-Effect(waves,devastation)* relation in the first dependency tree. However, the same node and edge are relatively irrelevant for the prediction of the *Instrument-Agency(entrance,contractors)* relation in the second dependency tree.

It is also worth mentioning that the dependencies between words in some cases are more important than the words themselves for relation classification. Consider the sentence (d) in Table 1. Replacing the word *assess* with another predicate will not change the *Instrument-Agency* relation between the two entities *analysts* and *frequency*. However, the dependency *nsubj* indicates the subject component of entity *analysts* and the dependency *advcl* indicates the adverbial component of the verb-object phrase containing entity *frequency*. As the syntactic component pair (object in adverbial component, subject) is highly consistent with the syntactic component pair of *Instrument* and *Agency*, these dependencies specifying the syntactic component pair is critical for the prediction of target relation. Based on this observation, the recent dependency tree-based approaches (Zhang, Qi, and Manning 2018; Guo, Zhang, and Lu 2019) without taking the dependency relation into consideration may not sufficiently capture the precise semantic information.

Based on these observations and issues discussed above, we are inspired to develop a general model which can distinguish the importance of the nodes (words) and edges (dependencies) and model information transport between connected nodes in the dependency tree automatically. In this paper, we propose a novel method to transform the dependency tree into a weighted graph which we refer to as a syntax-transport graph. The syntax-transport graph models the importance of nodes and edges by assigning a particular weight to all nodes and edges. To model the information transport of nodes in the syntax-transport graph, we further propose a multi-layer convolutional network which can operate on the syntax-transport graph directly. In the proposed model, the information of nodes is strictly guided by the syntax-transport graph. The information transport is biased to edges with large weights, and is restricted to edges with small weights. After all, this information transport is a strategy for composing information of nodes. Thus, each node in the syntax-transport graph is enhanced by composing the information from other nodes along the edges. Finally, we exploit a max-pooling on all nodes to distill a representation which potentially aggregates all the relevant information for relation classification.

The contribution of this paper is summarized as follow:

- We proposed a syntax-transport graph to distinguish the importance of nodes and edges in the dependency tree.
- We further introduced the LST-AGCN operating directly on the syntax-transport graph to model the information transport of nodes in the syntax-transport graph.
- The empirical study on Semeval-2010 Task 8 and Tacred confirmed the effectiveness of the proposed model.

Related Work

In earlier relation extraction studies, researchers focused on leveraging various kinds of linguistic features and manually designed feature in the task. However, all the feature-based methods depend strongly on the quality of designed features from a pre-processing step.

Most recent works have focused on leveraging neural networks in this task. The existing approaches can be categorized into two classes: sequence-based and dependency-based. The sequence-based models only make use of the sequence of the words. (Zeng et al. 2014) first exploits a convolutional neural network with manual designed features to encode relations. (Zhang et al. 2015) propose a bidirectional long short-term memory network (BLSTM) to model the sentence with complete, sequential information about all words. (Zhou et al. 2016) and (Shen and Huang 2016) proposed to leverage attention mechanism over RNN and CNN based models for this task. Apart from only using words sequence, the dependency-based models attempt to integrate the dependency tree into models. The dependency-based models have been shown to improve the performance by capturing long-distance relations. Specifically, the shortest dependency path (SDP) between entities in the dependency is believed to convey informative clues for target relation. (Xu et al. 2015) apply a LSTM over the SDP between entities. (Cai, Zhang, and Wang 2016) proposed a recurrent

convolutional neural network to encode the global pattern in SDP using a two-channel LSTM, and employ a CNN to capture the local features of every two neighbor words linked by a dependency relation.

However, the shortest dependency path may omit some important information about relations. Recent works have focused on pruning irrelevant information from the dependency tree. (Miwa and Bansal 2016) reduces the full tree to the subtree below the lowest common ancestor (LCA) of the entities. (Zhang, Qi, and Manning 2018) apply a graph convolutional networks (GCNs) (Kipf and Welling 2017) to model over a pruned tree. This tree includes tokens that are up to distance K away from the dependency path in the LCA subtree. However, the pruning strategies in their approaches is predefined and fixed. Such a rule-based pruning strategy is not general for most cases. (Guo, Zhang, and Lu 2019) employs a self-attention mechanism to transform the dependency tree into a fully connected weighted graph. The Attention Guided Graph Convolutional Networks (AGGCNs) is further proposed to operate on the graph to model the relations between entities. As all nodes are connected in their graph, the newly introduced edges may damage the structure of the dependency tree. How to effectively remove the irrelevant information from the dependency tree still remains a challenge in this task.

Problem Statement

In this section, we formally describe the relation classification problem. For a given sentence $s = \{w_1, w_2, \cdots, w_n\}$ with n words, entity $e = \{w_{p+1}, w_{p+2}, \cdots, w_{p+n_e}\}$ is a word phrase contained in s given by starting position index p and ending position index $p + n_e$, where n_e is the word length of entity e. Given two entities e_1 and e_2 of sentence s, the relation classification problem is to predict their relation $t(e_1, e_2) \in \mathbb{T}$, where $\mathbb{T} = \{t_1, \cdots, t_m\}$ is the set of all m different types of entity relation. Note that $t(e_1, e_2)$ and $t(e_2, e_1)$ commonly belong to two different relations.

Model

In this section, we gradually describe the proposed model. Firstly, we describe the framework of the syntax-transport GCN (ST-GCN). Nextly, we introduce the learnable syntax-transport GCN (LST-GCN). Then, we present the learnable syntax-transport attention GCN (LST-AGCN). Finally, we show the overall model for relation classification.

Syntax-transport GCN

The overview of ST-GCN is shown in Figure 2. The model directly operates over the dependency tree of a sentence. Generally, the dependency tree can be equivalently interpreted as a directed self-loop graph G, where nodes represent words in the sentence, and edges represent syntactic dependency paths between words in the dependency tree. And the graph G can be represented as an $n \times n$ adjacent matrix A, where n denotes the length of the sentence, with entities A_{ij} signaling if node i is connected to node j by a single dependency path in G. Specifically, $A_{ij} = 1$ if node i is connected to node j, and $A_{ij} = 0$ otherwise.

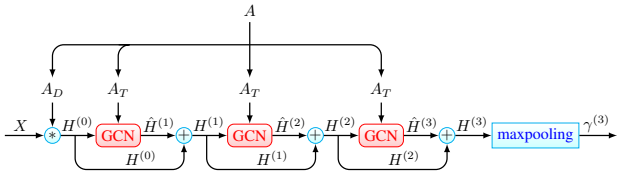

Figure 2: Overview of ST-GCN.

In this model, we represent the dependency tree using a weighted graph referred to as the syntax-transport graph. The syntax-transport graph describes the information transport among nodes in the dependency tree. Nodes and edges are given particular weights in this graph. In this way, the corresponding A representing this graph is not a binary matrix. Each entry A_{ij} is taken from $[0, 1]$ representing the percentage of information transported from node j to node i. Specifically, the diagonal element A_{ii} represents the importance of node i in the dependency tree, in other words, the percentage of information to be reserved in node i.

Given the matrix A, together with node embeddings $X = \{x_1, x_2, \ldots, x_n\}$, a GCN (Kipf and Welling 2017) can be operated directly on this graph to model the information transport of nodes. In a single layer GCN, nodes only get the information from their first-order neighborhood. For the K layer GCN, nodes get information from up to the K-th order neighborhood. At the k-th convolution layer of GCN, the embedding $h_i^{(k)}$ of a single node i updates as follows:

$$h_i^{(k)} = \phi\left(\sum_{j=1}^{n} c_i A_{ij}\left(W^{(k)} h_j^{(k-1)} + b^{(k)}\right)\right), \quad (1)$$

where $k = 1, 2, ..., K$,$h_j^{(k)}$ is the embedding of node j at the k^{th} layer, $h_j^{(0)} = x_j$, $b^{(k)}$ is a bias term, $W^{(k)}$ is a weight matrix. c_i is a normalization constant, which we choose as $c_i = 1/\sum_{j=1}^{n} A_{ij}$. $\phi(\cdot)$ denotes the ReLU activation function.

Roughly speaking, the ST-GCN is a multi-layer soft-gating GCN over the dependency graph. One challenge of the soft-gating mechanism in practice is that since $A_{ii} < 1$, the new node representations tend to become smaller as the layers deepen, especially when the nodes of the dependency tree have only one or two children. Besides, it is natural to consider the importance and the transport of node separably. To handle this problem, we weigh the importance of the node only once and gradually integrate its neighbor (i.e. children) information through multi GCN layers. Let $A_{\mathrm{D}} = \mathrm{diag}(A)$ be the diagonal matrix with diagonal elements same with A. $A_{\mathrm{T}} = A - A_{\mathrm{D}}$ be the adjacent matrix for the GCN. Formally, $H^{(k)}$, the output of the k-th GCN layer is computed as follows:

$$H^{(0)} = A_{\mathrm{D}} X \quad (2)$$

$$\hat{H}^{(k)} = \sigma(A_{\mathrm{T}} H^{(k-1)} W^{(k)}) \quad (3)$$

$$H^{(k)} = H^{(k-1)} + \hat{H}^{(k)}, \quad k = 1, 2, ..., K \quad (4)$$

where $W^{(k)}$ is the parameter in the k-th GCN layer, K is the number of layers.

After Eq. (2), nodes with large weights are reserved while nodes with small weights are removed. The residual nodes are informative for the classification task. In particular, the GCN can be visualized as an information propagation network which transforms and propagates information along the edges of the graph to update the node embeddings.

To the end of multi-layer GCN, a max-pooling operation is performed on the output of the last GCN layer $H^{(K)}$ to distill a representation that has potentially aggregated all the information of the nodes sufficient for classification.

$$\gamma^{(K)} = \text{max-pooling}(H^{(K)}) \tag{5}$$

Learnable Syntax-transport GCN

Recall that the adjacency matrix A is assumed to be given above. In this section, we describe the computation of A which now we refer to as the transporting matrix. Due to the fact that the transporting matrix A is learnable, we refer to this model as the learnable syntax-transport GCN (LST-GCN).

The LST-GCN models over the sentence dependency tree, which is directed and acyclic. In this tree, the verb word is often taken to be the structural center (the root of the tree) and other words are either directly or indirectly connected to the root. Considering two words w_i, w_j and their dependency relation r_{ij}, denoted as $w_i \xrightarrow{r_{ij}} w_j$, w_i is the father node of w_j. w_j is also called the dependent connected to its head w_i in terms of the dependency r_{ij}.

In the LST-GCN, we propagate all the dependents' information to its head along their syntax dependencies and choose a node that has potentially aggregated all the relevant information for relation classification. Therefore, we train a learnable syntax-transport graph which is represented by the transporting matrix A. Recall that A_{ij} implies how much information of node j will be transported to node i. It should relate to the two words w_i, w_j and the dependency relation r_{ij}. Considering that w_i, w_j and r_{ij} are only the local dependency features of node j, it may be insufficient to determine the information transport from node j to node i. So we introduce the contextual feature of node j as a supplement to computing A_{ij}. For all the dependency relations, $P = \{p_1, p_2, ..., p_l\}$ is their embedding matrix, where p_i is the vector representation of the i-th dependency relation. P is randomly initialized and updated together with other model parameters. The embeddings of words are denoted as $E = \{e_1, e_2, ..., e_N\}$.

We model A_{ij} by word embeddings e_i, e_j, dependency relation embedding $p_{r_{ij}}$, and the dependent node embedding o_j which is a BiLSTM embedding for node j.

$$A_{ij} = \sigma(w_{\mathrm{e}}^T(w_{\mathrm{h}}e_i + W_{\mathrm{d}}e_j + W_{\mathrm{p}}p_{r_{ij}}) + w_{\mathrm{o}}^T o_j) \tag{6}$$

where $W_{\mathrm{A}} = \{w_{\mathrm{h}}, W_{\mathrm{d}}, W_{\mathrm{p}}, w_{\mathrm{e}}, w_{\mathrm{o}}\}$ are trainable model parameters and sigmoid activation function σ is used. In this way, A_{ij} will be assigned to a real value taken from $[0, 1]$ when there is a directed link from i to j in the dependency tree, otherwise it will be zero.

Recall that the diagonal element A_{ii} of the transporting matrix in GCN implies how much the information of node i should be retained to its new representation. To unifying the calculation of A, we introduce a special "self-loop dependency relation" r_{l}, and add the dependency $w_i \xrightarrow{r_{\mathrm{l}}} w_i$ to all nodes in the dependency tree. Thus, all the diagonal elements of A are also learned according to Eq. (6).

Learnable Syntax-transport Attention GCN

Another challenge in the multi-layer model is to determine the number of layers K. Theoretically, if the transporting matrix A is given, we can easily choose the optimal K according to A. However, the transporting matrix A is learnable in our model. It is impossible to choose a fixed K optimal to the transporting matrix A which is varying during the training process. In addition, a fixed K is not optimal for all the instances in the dataset.

Taking these issues into account, we exploit a max pooling-operation on the output of each GCN layer:

$$\gamma^{(k)} = \text{max-pooling}(H^{(k)}), \quad k = 1, 2, ..., K \tag{7}$$

Then, we adopt attention mechanism to soft-choose all representations $H = \{\gamma^{(1)}, \cdots, \gamma^{(K)}\}$ output from all the layers. The weighted combination of all the K layers is then the final representation of a sentence s for relation classification.

$$v_s = \text{softmax}((Hw_{\mathrm{h}})^T)H \tag{8}$$

where w_{h} is the attention parameters to be trained. We refer to this model as a learnable syntax-transport attention GCN model (LST-AGCN). Overview of LST-AGCN is shown in Figure 3.

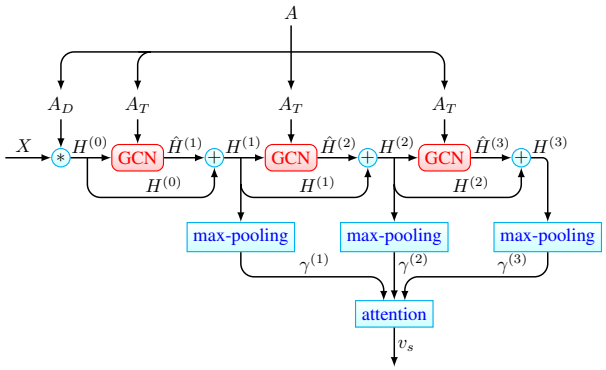

Figure 3: Overview of LST-AGCN.

Overall Model for Relation Classification

In this LST-AGCN model, each node in the dependency tree is encoded into a high dimensional embedding. For a given sentence $s = \{w_1, w_2, \cdots, w_n\}$ with n words, we perform a lookup in the embedding matrix to obtain the vector representation $E = \{e_1, e_2, \cdots, e_N\}$ of words. The embedding matrix is the vector representation of all words in the dictionary. We use the pre-trained GloVe embeddings (Pennington, Socher, and Manning 2014) and update it together with the other model parameters. Besides, special position symbols are used to mark the entities in the sentence. We add the "position symbols" to the word dictionary and randomly

initialize their embeddings. To further encode the contextual information into word embeddings, we exploit a BiLSTM on E. For each word embedding e_i, the BiLSTM returns a forward representation $\overrightarrow{x_i}$ and a backward representation $\overleftarrow{x_i}$. The concatenation $x_i = [\overrightarrow{x_i}, \overleftarrow{x_i}]$ is our final node embeddings, namely $X = \{x_i\}|_{i=0}^{n}$.

Finally, we use a softmax classifier to predict label $\hat{y_s}$ for the sentence s, and for all the sentence s and the target relation label $\hat{t_s}$, we minimize the cross-entropy loss between the true relation label distribution and the predicted label distribution over all the model parameters including word embedding matrix, dependency relation embedding matrix P, weight parameters W_{A}, GCN weight parameters $W^{(1)}, \cdots, W^{(K)}$, attention weight parameters w_{h} and predictive possibility weight parameter W_{c}.

$$y_s = \mathrm{softmax}(v_s W_{\mathrm{c}} + b_{\mathrm{c}}) \quad (9)$$

$$\mathcal{L} = \sum_{s} \left(-\frac{1}{m} \sum_{i=1}^{m} t_s^i \log(y_s^i) \right) \quad (10)$$

where $t_s \in \mathbb{R}^m$ is the one-hot represented ground truth, $y_s^i \in \mathbb{R}$ is the estimated probability for the ith class, m is the number of target classes.

Experiment

In this section, we conduct experiments to validate our model on benchmark datasets. Specifically, we perform experiments on the Semeval-2010 Task 8 (Hendrickx et al. 2010) (Semeval) and the Tacred (Zhang et al. 2017) datasets. For Semeval, we use the macro-averaged F1 score, which is officially used as the evaluation metric. For Tacred, we use the micro-averaged F1 score, which is the main evaluation metric used on the dataset. We summarize the statistics of the datasets in Table 2.

Dataset	Train	Dev	Test
Semeval	7200	800	2717
Tacred	68124	22631	15509

Table 2: Distribution of splits on benchmark datasets.

Implementation and Parameter Settings

We exploit 300-dimensional Glove vectors (Pennington, Socher, and Manning 2014) for the word embeddings, as well as a 30-dimensional part-of-speech (POS) embeddings, 30-dimensional named entity recognition (NER) embeddings, and 30-dimensional dependency relation (DEP) embeddings. We concatenate both word, POS and NER embeddings, and learn a 300-dimensional BiLSTM embeddings for each word. We randomly dropout 10% of neurons in the first GCN layer, and 10% of neurons in the input layer. Our model is trained for 100 epochs with batch size 50. We use the SGD optimizer with an initial learning rate of 0.7 for all datasets. The sentences in Semeval-2010 Task 8 are parsed by the Stanford parser [1].

[1] https://stanfordnlp.github.io/CoreNLP/

Performance Comparison

We now show the results on Semeval and Tacred dataset. As a baseline, we include the SVM classifier (SVM) (Rink and Harabagiu 2010), Shortest path LSTM (SDP-LSTM) (Xu et al. 2015), tree structural neural network (SPTree) (Miwa and Bansal 2016), Position Aware LSTM (PA-LSTM) (Zhang et al. 2017). We mainly compare our model with C-GCN (Zhang, Qi, and Manning 2018) and C-AGGCN (Guo, Zhang, and Lu 2019). Both models exploit GCN to operate on the pruned dependency trees. All the results of the baseline models are taken from (Guo, Zhang, and Lu 2019). For evaluating the effectiveness of our proposed model, we also show the results of ST-GCN and LST-GCN. Note that in the ST-GCN model, we use the binary adjacent matrix as the transporting matrix A and a 3 layer GCN is employed.

Model	F1
SVM (Rink and Harabagiu 2010)	82.2
SDP-LSTM (Xu et al. 2015)	83.7
SPTree (Miwa and Bansal 2016)	84.4
PA-LSTM (Zhang et al. 2017)	82.7
C-GCN (Zhang, Qi, and Manning 2018)	84.8
C-AGGCN (Guo, Zhang, and Lu 2019)	85.7
ST-GCN	85.0
LST-GCN ($K = 1$)	84.7
LST-GCN ($K = 2$)	84.9
LST-GCN ($K = 3$)	85.5
LST-GCN ($K = 4$)	85.0
LST-AGCN ($K = 3$)	**86.0**

Table 3: Performance comparison of different models on the Semeval-2010 Task 8 dataset. The best performance is bold-typed. K represents the number of GCN layers.

Model Evaluation on Semeval. Table 3 shows the performances from various baseline models. It can be seen that LST-AGCN ($K = 3$) outperforms all previous models. Specifically, both our model LST-AGCN and C-AGGCN significantly surpasses C-GCN, revealing the superiority of 'soft pruning' methods over the 'hard-rule pruning' methods. Even with simple architectures, we find that ST-GCN has outperformed most of the baseline models and shows competitive performance with the state-of-the-art C-AGGCN. By making the transporting matrix A learnable, the LST-GCN ($K = 3$) has only a degradation of 0.2 compared to C-AGGCN. LST-AGCN ($K = 3$) which exploits an attention mechanism achieves a performance improvement of 0.3 over C-AGGCN.

Model Evaluation on Tacred. We also conduct experiments on Tacred dataset. We compare our proposed methods with the baseline methods and present the results in table 4. It can be observed that our proposed method LST-GCN for $K = 1, 2, 3$ significantly outperforms the baseline methods, except the reported result of C-AGGCN. However, we returned to the source code provided by (Guo, Zhang, and Lu 2019) to reproduce the result reported for C-AGGCN. The best result we could obtain was 67.7. (Guo, Zhang, and Lu 2019) have claimed to revise the results in their paper, publishing a stable result of about 68.2 for C-AGGCN. We find

that our model LST-AGCN with $K = 2$ shows competitive performance with C-AGGCN.

Result Discussion For the LST-GCN model, it can be noted that the macro-F1 and micro-F1 performance increases with increasing GCN layers in both Semeval and Tacred respectively. However, we find out the model encounters a performance degradation after a number of layers. As elaborated in the model section, we cannot predetermine a fixed K optimal to the learnable transporting matrix A. A fixed K might not be optimal for all instances as well. For the performance deterioration after a number of layers, a possible explanation is that deep GCNs operating on shallow dependency trees tend to over-smooth node representations, making node representations indistinguishable and thereby hurting the model performance (Li, Han, and Wu 2018). We conduct experiments to further study this behavior. To deal with the limitation of LST-GCN, the LST-AGCN has been proposed to soft-choose all representations output from all GCN layers. We find that LST-AGCN achieves a performance improvement of 0.5 for $K = 3$ on the Semeval and a performance improvement of 0.2 on the Tacred. Notice that choosing K for LST-AGCN results in a trade-off between the avoidance of node smoothening and soft-choosing rich representations. Thus, care must be taken when choosing K for LST-AGCN.

Model	F1
LR (Zhang et al. 2017)	59.4
SDP-LSTM (Xu et al. 2015)	58.7
Tree-LSTM (Tai, Socher, and Manning 2015)	62.4
PA-LSTM (Zhang et al. 2017)	65.1
C-GCN (Zhang, Qi, and Manning 2018)	66.4
C-AGGCN (Guo, Zhang, and Lu 2019)	**69.0** (67.7*)
ST-GCN	67.2
LST-GCN ($K = 1$)	68.3
LST-GCN ($K = 2$)	68.6
LST-GCN ($K = 3$)	66.6
LST-GCN ($K = 4$)	65.9
LST-AGCN ($K = 2$)	**68.8**

Table 4: Performance comparison of different models on the Tacred dataset. The best performance is in bold. K represents the number of GCN layers. "*" marks the best result produced from rerunning the published source code.

Ablation Study

To examine the contributions of the main model components, we conduct ablation experiments on the Semeval and Tacred datasets. We focus on the LST-AGCN with 3 GCN layers for Semeval and 2 GCN layer for Tacred. Table 5 shows the results of our experiments. The ablated models are characterized by (1) LST-AGCN$_{-A_D}$: a LST-AGCN with A_D set to the identity matrix, ensuring equal importance on all nodes, (2) LST-AGCN$_{-A_T}$: a LST-AGCN with A_T set to the binary adjacency matrix, ensuring node information is propagated equally, (3) LST-AGCN$_{-\text{Attention}}$: a LST-AGCN which has been reduced to a LST-GCN (4) LST-AGCN$_{-\text{GCN}}$: a LST-AGCN which does not utilize the GCN but simply performs a max-pooling operation on the BiLSTM embeddings of the nodes to distill a representation.

Model	Semeval	Tacred
LST-AGCN	86.0	68.8
LST-AGCN$_{-A_D}$	84.7	66.7
LST-AGCN$_{-A_T}$	84.9	67.8
LST-AGCN$_{-\text{Attention}}$	85.5	68.6
LST-AGCN$_{-\text{GCN}}$	84.6	57.1

Table 5: Performance of different ablation models of LST-AGCN.

We find that the performance of LST-AGCN deteriorates as we remove critical components. Specifically, LST-AGCN$_{-A_D}$ underperforms relative to LST-AGCN, suggesting that the importance of nodes must be modeled for performance improvement. We also find that LST-AGCN$_{-A_T}$ underperforms relative to LST-AGCN, giving an indication on the fact that information being propagated from a node to the K-th neighbor nodes must be weighed based on the importance of nodes. LST-AGCN outperforms LST-AGCN$_{-\text{Attention}}$ (LST-GCN model), which is expected from our explanation earlier. We also find that LST-AGCN significantly outperforms LST-AGCN$_{-\text{GCN}}$. The results suggest that GCNs improve BiLSTM embeddings. Moreover, in Tacred which has lots of lengthy sentences with multiple entity pairs, we find that the performance significantly drops. It seems that the GCN is efficient to model the dependency structure of a sentence to properly distinguish multiple entity relations.

Case Study

In this section, we present a visualization of the behavior of LST-AGCN on three instances chosen from Table 1, with the aim to validate our motivation provided in the introduction section. We wish to examine whether LST-AGCN indeed assigns high weights to important words and structures in the transporting matrix A through a per instance inspection. The visualization result is shown in Figure 4.

In Figure 4, we find that for the instance containing the two entity pairs *(man,support pillow)* and *(pain,stresses)*, LST-AGCN is able to distinguish the relevant substructures according to the given entity pair. This is seen by the high weights assigned to word pairs in the transporting matrix or the highlighted edges.

Consider the second instance with entity pair *(entrance,contractors)* and third instance with entity pair *(frequency,Analysts)*. We find that in the second instance, the node *by* is significant to indicate the subject-predicate relation between the two entities. However, in the third instance, *by* is relatively irrelevant with the classification of the target relation *Instrument-Agency(entrance,contractors)*. LST-AGCN is able to identify the relevance of the word *by* in terms of the context to properly learn the correct relational features. We see this by the high weight it assigns to the third instance and the low weight on the second instance in the transporting matrix.

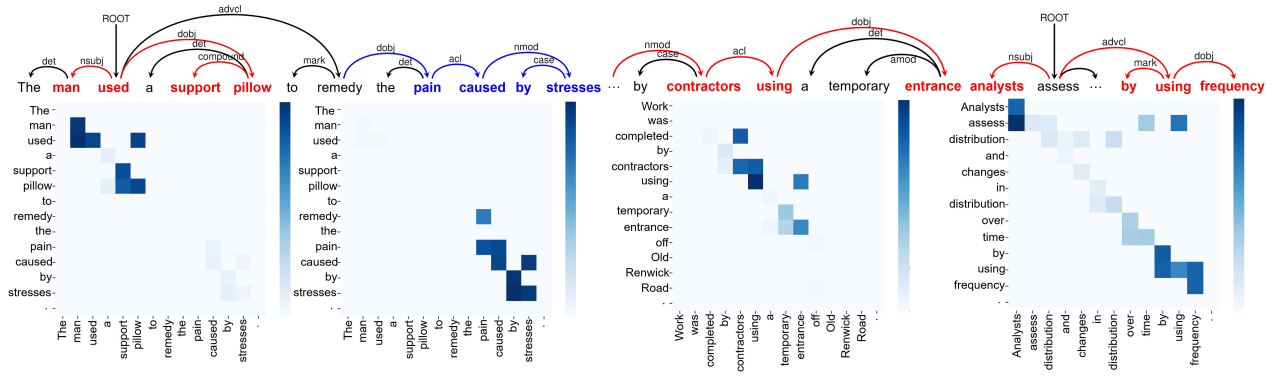

Figure 4: Visualization of the transporting matrix A in LST-AGCN. The highlighted ($A_{ij} > 0.5$) nodes together with the highlighted edges are defined as the relevant substructure. The first two instances are chosen from a single sentence. The relevant substructures are marked in red and blue respectively. The last two instances are chosen from different sentences. The entity relation of these four instances are *Instrument-Agency(support pillow, man)*, *Cause-Effect(stresses,pain)*, *Instrument-Agency(entrance,contractors)* and *Instrument-Agency(frequency,Analysts)*.

	Depth(Prop)	Depth(Prop)	Depth(Prop)	Depth(Prop)	Depth(Prop)
Model	2(14.28%)	3(24.9%)	4(25.36%)	5(18.66%)	6(9.31%)
LST-GCN ($K = 1$)	6.19%	11.82%	11.03%	9.66%	8.30%
LST-GCN ($K = 2$)	6.44%	11.67%	11.90%	9.27%	8.70%
LST-GCN ($K = 3$)	6.96%	12.41%	11.03%	9.27%	7.51%
LST-AGCN ($K = 3$)	6.44%	11.08%	11.47%	8.88%	7.51%

Table 6: Error rate of LST-GCN and LST-AGCN on instances with different dependency tree depth, the "Other" relation is excluded. "Prop" denotes the proportion of total instances having a particular depth.

The Impact of Attention Mechanism in LST-AGCN

As discussed above, we speculate that a fixed K for the GCN layer LST-GCN is not optimal for most cases, and a relatively large K hurts performance. Here, we present experimental results to buttress this claim and show that the attention mechanism of LST-AGCN solves the problem. Experimental results are shown in Table 6. The results indicate that LST-GCN is sensitive to K and the depth of dependency trees. Specifically, LST-GCN shows increasing error rates for increasing K on shallow dependency trees. Also, LST-GCN shows decreasing error rates for increasing K on deep dependency trees. This is due to the over-smoothing problem of deep GCNs and the fact that the transporting matrix is learnable as explained earlier. Hence, an optimal K cannot be chosen for LST-GCN. However, LST-AGCN can improve upon the performance through soft-choosing representations distilled by LST-GCN layers. We find that LST-AGCN achieves low error rates for different depths of the dependency tree. The result confirms our claim and show the effectiveness of the attention mechanism exploited in LST-AGCN.

Conclusion

Relation classification is a basic block of natural language processing tasks. While most recent work leverage dependency grammar information of sentences, the effect of the tree structure and the dependency relations in the dependency tree is not fully explored. In this paper, we proposed a learnable attention graph convolutional network over the syntax-transport graph (LST-AGCN) for relation classification. In this model, both the word features and syntax relation information are transported and aggregated according to the grammar structure. And with attention mechanism, different levels of propagation are softly mixed in the final representation for the classification task. Experimental results on public dataset Semeval-2010 and Tacerd verified the effectiveness of our proposed approach and showed that it outperforms the state-of-the-art in terms of classification accuracy. As our work depends on the dependency tree, the quality of the parsing result may affect the performance of our model. In the future, we will further study the behavior of our model with different parsing tools.

Acknowledgments

This work is supported partly by the National Natural Science Foundation of China (No. 61772059, 61421003), by the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), by State Key Laboratory of Software Development Environment (No. SKLSDE-2018ZX-17), by the Beijing S&T Committee (No. Z191100008619007) and by the Fundamental Research Funds for the Central Universities.

References

Cai, R.; Zhang, X.; and Wang, H. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Guo, Z.; Zhang, Y.; and Lu, W. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, 241–251.

Hendrickx, I.; Kim, S. N.; Kozareva, Z.; Nakov, P.; Séaghdha, D. Ó.; Padó, S.; Pennacchiotti, M.; Romano, L.; and Szpakowicz, S. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, 33–38.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Lee, J.; Seo, S.; and Choi, Y. S. 2019. Semantic relation classification via bidirectional LSTM networks with entity-aware attention using latent entity typing. *Symmetry* 11(6):785.

Li, Q.; Han, Z.; and Wu, X. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 3538–3545.

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

Miwa, M., and Bansal, M. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Rink, B., and Harabagiu, S. M. 2010. UTD: classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, 256–259.

Shen, Y., and Huang, X. 2016. Attention-based convolutional neural network for semantic relation extraction. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, 2526–2536.

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

Xu, Y.; Mou, L.; Li, G.; Chen, Y.; Peng, H.; and Jin, Z. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, 1785–1794.

Yang, Y.; Tong, Y.; Ma, S.; and Deng, Z. 2016. A position encoding convolutional neural network based on dependency tree for relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, 65–74.

Zeng, D.; Liu, K.; Lai, S.; Zhou, G.; and Zhao, J. 2014. Relation classification via convolutional deep neural network. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, 2335–2344.

Zhang, S.; Zheng, D.; Hu, X.; and Yang, M. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation, PACLIC 29, Shanghai, China, October 30 - November 1, 2015*.

Zhang, Y.; Zhong, V.; Chen, D.; Angeli, G.; and Manning, C. D. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, 35–45.

Zhang, Y.; Qi, P.; and Manning, C. D. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, 2205–2215.

Zhou, P.; Shi, W.; Tian, J.; Qi, Z.; Li, B.; Hao, H.; and Xu, B. 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, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.