# Learning to Prune Dependency Trees with Rethinking for Neural Relation Extraction

**Bowen Yu**[1,2], **Mengge Xue**[1,2], **Zhenyu Zhang**[1,2],
**Tingwen Liu**[1,2]*, **Yubin Wang**[1,2] and **Bin Wang**[3]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Xiaomi AI Lab, Xiaomi Inc., Beijing, China
{yubowen,xuemengge,zhangzhenyu1996}@iie.ac.cn
{liutingwen,wangyubin}@iie.ac.cn wangbin11@xiaomi.com

## Abstract

Dependency trees have been shown to be effective in capturing long-range relations between target entities. Nevertheless, how to selectively emphasize target-relevant information and remove irrelevant content from the tree is still an open problem. Existing approaches employing pre-defined rules to eliminate noise may not always yield optimal results due to the complexity and variability of natural language. In this paper, we present a novel architecture named Dynamically Pruned Graph Convolutional Network (DP-GCN), which learns to prune the dependency tree with rethinking in an end-to-end scheme. In each layer of DP-GCN, we employ a selection module to concentrate on nodes expressing the target relation by a set of binary gates and then augment the pruned tree with a pruned semantic graph to ensure the connectivity. After that, we introduce a rethinking mechanism to guide and refine the pruning operation by feeding back the high-level learned features repeatedly. Extensive experimental results demonstrate that our model achieves impressive performance compared to strong competitors.

## 1 Introduction

Relation extraction (RE) aims to detect the semantic relationship between two specific entities appearing in a sentence (often termed subject and object, respectively). This task plays an important role in many downstream NLP applications that require a relational understanding of unstructured text such as question answering (Dai et al., 2016) and dialogue systems (Young et al., 2018).

Models leveraging the dependency tree of the input sentence have proven to be effective in relation extraction because they can effortlessly exploit long-term relations that are obscure from the surface form (Zhang et al., 2018; Can et al., 2019). Recent studies also stated that not all tokens in the dependency tree are needed to express the relation of the target entity pair (Xu et al., 2015b; Zhang et al., 2018), and some target-irrelevant tokens could introduce noise and cause confusion to the classification. Therefore, multiple pruning strategies are proposed to eliminate unimportant tokens and distill the dependency information. Xu et al. (2015b) applied neural networks only on the shortest dependency path (SDP) between the two entities in the dependency tree, which soon became dominant with many works demonstrating that using SDP brings better experimental results than using the whole sentence (Xu et al., 2015a; Cai et al., 2016). Miwa and Bansal (2016) reduced the full tree to the subtree below the lowest common ancestor (LCA) of the entities. Zhang et al. (2018) expanded SDP by including tokens that are up to distance $K$ away from the dependency path in the LCA subtree. However, these hand-crafted pruning rules may lead to the omission of useful information due to the variability and ambiguity of natural language. Look at a concrete example shown in Figure 1, the key relational token "Founded" is always excluded from the pruned tree no matter what kind of pruning rule mentioned above is deployed. In fact, it's unrealistic to expect an empirical rule to deal with all situations, and an ideal dependency-based

*Corresponding author.

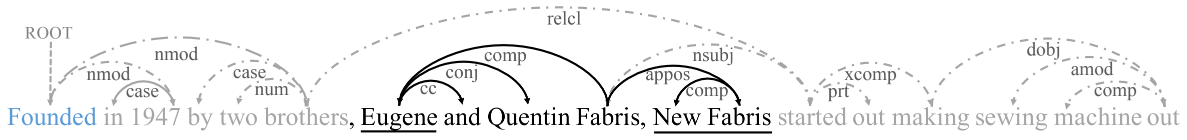
ROOT
nmod
nmod
case
case
num
relcl
comp
conj
cc
nsubj
appos
comp
xcomp
prt
dobj
amod
comp
Founded in 1947 by two brothers, Eugene and Quentin Fabris, New Fabris started out making sewing machine out

Figure 1: An example dependency tree expressing a relation *org:founded-by* between "Eugene" and "New Fabris". Note that even the most relaxed pruning rule LCA subtree (highlighted in bold) still excludes the relational token "Founded", resulting in the loss of crucial information.

model should be able to learn how to remove irrelevant information from the tree while keeping relevant content for the specific entity pair to the greatest extent.

In this paper, we propose a novel architecture named **D**ynamically **P**runed **G**raph **C**onvolutional **N**etwork (DP-GCN), which takes the full dependency tree as input and learns to prune the tree with rethinking in an end-to-end training manner. At the heart of DP-GCN is a selection module that dynamically identifies a subset of critical nodes in the dependency tree that provide sufficient information to extract the relation between two entities. This module takes into account the semantics of each node and the target entities and generates a set of input-dependent binary gates to determine whether each node should be kept. One problem coming with dynamic pruning is that selecting sub-structure from the dependency tree directly may result in a disconnected topology because the dependency tree is sparse, which hinders the message propagation between nodes. To address this issue, we enhance the pruned tree with a pruned semantic graph generated by the self-attention mechanism. Then on top of the resulting graph, a GCN module (Kipf and Welling, 2016) is exploited to update the entity-specific context representations, and such a prune-then-update process can be stacked over $L$ layers. Furthermore, instead of pruning the tree based on one-pass of the data through the network, we introduce feedback connections and endow the network with the ability to "rethink" the pruning operation by transferring the high-level features into the selection module of each layer. Benefiting from the rethinking mechanism, the model is able to reselect nodes with consideration of previous pruning information and extract more discriminative target-specific features with the guidance from the high-level semantic.

To summarize, our contributions are three-fold:

- We propose a novel dynamically pruned graph convolutional network for relation extraction, which is capable of pruning the dependency tree for the target entities without relying on pre-defined rules.
- We introduce a rethinking mechanism to enhance the pruning ability by leveraging the high-level feedback semantic to guide and refine the pruning operation.
- Experiments conducted on two public datasets show that our model consistently achieves superior performance over previous competing approaches. Extensive validation studies demonstrate the effectiveness of our pruning with rethinking method.

## 2 Related Work

Our work is inspired by two lines of research: enhancing relation extractor through syntactic dependency information and refining neural network with the rethinking mechanism.

**Dependency-based relation extraction.** Syntactic dependency information has been widely explored in relation extraction approaches for many years. Some early works introduced syntactic features into statistical classifiers and found them to be beneficial (Zelenko et al., 2003). Instead of using the full dependency tree, Bunescu and Mooney (2005) observed that the information relevant to relation extraction is almost entirely concentrated in the shortest dependency path (SDP) between two entities, and designed the dependency path kernel based on the SDP features. Based on the idea that SDP contains essential information, many studies exploited it with several refinements. Ebrahimi and Dou (2015) modified the original recursive neural network (RecNN) and presented an SDP-based RecNN for relation classification. Xu et al. (2015a) proposed to learn relation representations from SDP through CNN. Xu et

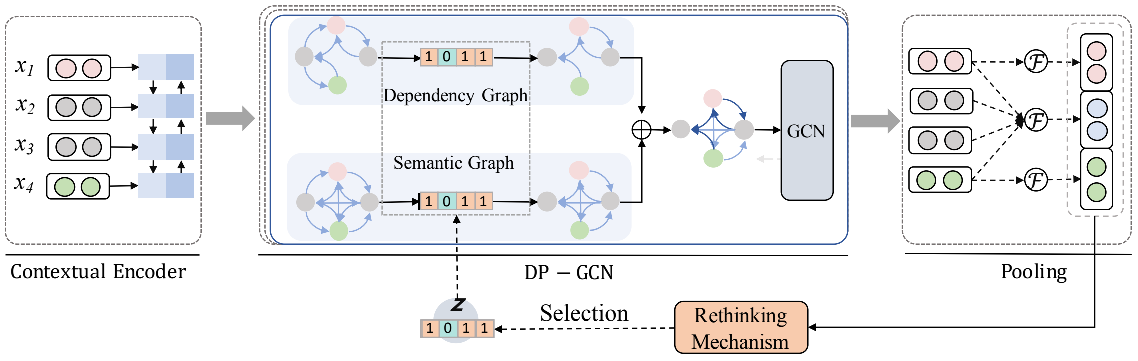

Figure 2: Model architecture (Best viewed in color). $x_1$ and $x_4$ denote subject entity and object entity, respectively. The model first encodes the contextual information, and then $L$ layers of DP-GCNs are deployed. In each layer, a selection module takes the node representations and the feedback high-level entity-specific features as input to select the relevant nodes and prune the dependency graph (self-loops are omitted for simplification). A pruned semantic graph generated by self-attention is also introduced to ensure the graph connectivity. Then the resulting graph is passed to a GCN module to propagate messages. Finally, a pooling module is leveraged to aggregate information. The obtained relational features are fed back to the selection module of each layer to adjust the pruning operation.

al. (2015a) designed a multi-channel LSTM to pick up heterogeneous information along with the SDP. Liu et al. (2015) augmented SDP with the subtrees attached to the shortest path, and utilized two neural networks to model the obtained structure. Cai et al. (2016) combined CNN and two-channel LSTM to make use of dependency relations information in the SDP. Miwa and Bansal (2016) found it to be effective when applying a Tree-LSTM to the subtree rooted at the lowest common ancestor (LCA) of the two entities. He et al. (2018) derived the context embedding of an entity over its dependency subtree in bottom-up order. Zhang et al. (2018) claimed that keeping only the SDP could lead to loss of crucial information and conversely hurt robustness, and proposed a path-centric pruning strategy to incorporate nodes that are directly attached to the path. Tran et al. (2019) built RNN on the SDP to gain long-distance features, which are combined with a CNN to preserve the full information. Unlike these methods that remove edges in preprocessing with hard rules, our model learns to prune the dependency tree in an end-to-end fashion. Recently, Guo et al. (2019) constructed a fully-connected graph for relation extraction via multi-head self-attention mechanism. Sun et al. (2020) proposed a learnable syntax-transport attention graph convolutional network which operates on the syntax-transport graph. However, they neglect the target entity information in the graph learning process and constructs a denser entity-unaware graph. In contrast, our approach not only constructs an entity-specific graph but also removes noisy information explicitly by the dynamic pruning strategy.

**Rethinking mechanism.** Previous attempts to use a rethinking mechanism in neural networks have been made in image classification (Li et al., 2018) and named entity recognition (Gui et al., 2019) to refine feature maps and tackle conflicts. We extend this concept to guide and adjust the pruning process based on the learned high-level semantic.

## 3 Methodology

The goal of our model is to predict the relationship between two entities in a given sentence. Figure 2 illustrates the overall architecture of the proposed model, which can be classified into three components : (1) The left panel is a BiLSTM encoder that transforms the input words into the contextualized representations. (2) The middle part, as the core of the whole model, contains $L$ layers of DP-GCNs, which incorporate entity information into the graph modeling process and filter useless information for the given entities. (3) The right panel is a pooling module used to aggregates node representations induced from the former DP-GCN layers. Next, we detail all components sequentially from left to right.

### 3.1 Contextual Encoder

Let $X = [x_1, ..., x_n]$ denote an $n$-word sentence, we embed each word token into a low-dimensional real-valued vector space with pre-trained embedding matrix. With the word embeddings of the sentence, a bidirectional LSTM is employed to produce hidden state vectors $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n]$,

$$\mathbf{h}_i = [\overrightarrow{\text{LSTM}}(\mathbf{x}_i); \overleftarrow{\text{LSTM}}(\mathbf{x}_i)], i \in [1, n], \tag{1}$$

where $\mathbf{h}_i \in \mathbb{R}^d$ represents the hidden state vector at time step $i$. In doing so, we can integrate contextual information in the word embeddings by keeping track of dependencies along the chain of words. Moreover, these representations are used as initial node features in the dependency tree.

### 3.2 Dynamically Pruned GCN

Formally, the dependency tree is a special form of graph $G$ with $n$ nodes, where nodes denote words in the sentence, and edges denote syntactic dependency paths between words in the graph. $G$ can be represented with an $n \times n$ adjacency matrix $\mathbf{A}$. If there is a dependency edge between words $i$ and $j$, then $\mathbf{A}_{ij} = 1$, and $\mathbf{A}_{ij} = 0$ otherwise. Following popular choice (Zhang et al., 2018), we also add a self-loop to each node and normalize $\mathbf{A}$ by the node degree. GCN is designed to deal with data containing graph structure. In an $L$-layer GCN, if we denote $\mathbf{h}_i^{l-1}$ the input state and $\mathbf{h}_i^l$ the output state of node $i$ at the $l$-th layer, the graph convolutional operation can be defined as:

$$\mathbf{h}_i^l = g(\sum_{j=1}^{n} \mathbf{A}_{ij} \mathbf{W}^l \mathbf{h}_j^{l-1} + \mathbf{b}^l), \tag{2}$$

where $\mathbf{W}^l \in \mathbb{R}^{d \times d}$ is a linear transformation, $\mathbf{b}^l \in \mathbb{R}^d$ is a bias term, and $g$ is a nonlinear activation function (e.g. ReLU). In this way, a node iteratively aggregates the information from its neighbors and updates the representation. However, as discussed in Section 1, directly using $\mathbf{A}$ as the input for relation extraction is not optimal because it contains many irrelevant nodes for the target entity pair. Therefore, at each layer $l$, we design a selection module to comprehend the entity-specific context and dynamically select out the crucial target-related nodes from the graph. This is achieved by introducing a set of binary gates $\{z_1^l, \cdots, z_n^l\}$, $z_i^l \in \{0, 1\}$ associated with each node. The $i$-th gate of the $l$-th layer is open when $z_i^l = 1$ and is closed when $z_i^l = 0$. It controls whether the information from the current node should be propagated and aggregated in the graph. Under this definition, we adapt $\mathbf{A}$ as follows:

$$\hat{\mathbf{A}}_{ij}^l = \frac{z_j^l \cdot \mathbf{A}_{ij}}{\epsilon + \sum_{m=1}^{n} z_m^l \cdot \mathbf{A}_{im}}, \tag{3}$$

where $\hat{\mathbf{A}}^l$ represents the pruned dependency matrix of the $l$-th layer, symbol $\cdot$ means multiplication, and a small quantity $\epsilon$ is added to the denominator to avoid numerical instabilities. For each node with closed gate ($z_j^l = 0$), we have $\hat{\mathbf{A}}_{*j}^l = 0$ and the corresponding hidden state $\mathbf{h}_j^{l-1}$ is not included into the aggregation of the $l$-th layer. Only selected nodes with open gates ($z_j^l = 1$) can pass messages to update the representations of other nodes and themselves. Unfortunately, it is difficult to guarantee that $\hat{\mathbf{A}}^l$ can be formulated as a graph, because deleting edges on the sparse dependency graph may separate the original graph into several disconnected ones, which is extremely unfavorable for GCN's message-passing process. To enhance the connectivity, inspired by (Guo et al., 2019), we augment $\hat{\mathbf{A}}^l$ with a graph constructed by the self-attention mechanism. The formulation can be written as:

$$\mathbf{E}^l = \frac{(\mathbf{K}^l \mathbf{W}_k^l)(\mathbf{Q}^l \mathbf{W}_q^l)^\top}{\sqrt{d}}, \tag{4}$$

$$\tilde{\mathbf{A}}_{ij}^l = \frac{z_j^l \cdot \exp(\mathbf{E}_{ij}^l)}{\sum_{m=1}^{n} z_m^l \cdot \exp(\mathbf{E}_{im}^l)}, \tag{5}$$

$$\bar{\mathbf{A}}_{ij}^l = \frac{\hat{\mathbf{A}}_{ij}^l + \tilde{\mathbf{A}}_{ij}^l}{2}, \tag{6}$$

where $\mathbf{E}^l_{ij}$ is the attention weight of edge from node $j$ to node $i$, $\mathbf{Q}^l$ and $\mathbf{K}^l$ are both equal to the collective representation $\mathbf{H}^{l-1}$ from the previous layer, $\mathbf{W}^l_k \in \mathbb{R}^{d\times d}$ and $\mathbf{W}^l_q \in \mathbb{R}^{d\times d}$ are trainable parameters for projection. These attention weights are normalized with the selection results to represent the relative importance. The obtained attention score matrix $\tilde{\mathbf{A}}^l$ can be considered as an adjacency matrix of a pruned semantic graph. Note that $\tilde{\mathbf{A}}^l$ is always connected unless all nodes are removed because $\mathbf{E}^l$ is fully-connected, so the augmented pruned dependency graph $\bar{\mathbf{A}}^l$ can satisfy the connectivity requirement. Then we apply a GCN module over $\bar{\mathbf{A}}^l$ to propagate message. Besides, we also employ residual connections to allow high-level networks to take the hidden states from low-level networks as additional input. That is, the output state of node $i$ at the $l$-th layer is $g(\sum_{j=1}^n \bar{\mathbf{A}}^l_{ij}\mathbf{W}^l\mathbf{h}^{l-1}_j + \mathbf{b}^l) + \mathbf{h}^{l-1}_i$. These connections serves as shortcuts that create a more closely coupled and efficient model.

In order to generate the binary gates, we build a semantic decision-making scheme that evaluates the contribution of each node for expressing the relationship between target entity pair by a set of probabilities $\{p^l_1, \cdots, p^l_n\}$. The detailed formula is given below:

$$p^l_i = \sigma(\mathbf{v}_l^\top \tanh(\mathbf{W}^l_h\mathbf{h}^{l-1}_i + \mathbf{W}^l_p[\mathbf{m}^l; \mathbf{s}^l; \mathbf{o}^l])), \tag{7}$$

$$p^l_{i,0} = 1 - p^l_i, \quad p^l_{i,1} = p^l_i, \tag{8}$$

$$z^l_i = f_{binarize}(p^l_i) = \arg\max_t p^l_{i,t},\ t = 0, 1. \tag{9}$$

Here $p^l_i$ determines the probability of the $i$-th node being selected at the $l$-th layer ($z^l_i = 1$), $f_{binarize} : [0,1] \to \{0,1\}$ binarizes the input value, $\mathbf{m}^l$ is a summary vector encoding information about the entire graph, $\mathbf{s}^l, \mathbf{o}^l \in \mathbb{R}^d$ stand for the representations of the subject and object, respectively, $[\cdot;\cdot]$ is the concatenation operator, $\mathbf{W}^l_h \in \mathbb{R}^{d\times d}, \mathbf{W}^l_p \in \mathbb{R}^{3d\times d}$ are parameter matrices, $\mathbf{v}_l \in \mathbb{R}^d$ is a context vector to be learned during training, $\sigma$ denotes the logistic sigmoid function, and the detailed calculation of $\mathbf{m}^l, \mathbf{s}^l, \mathbf{o}^l$ will be described in Section 3.4. We implement $f_{binarize}$ as a deterministic step function $z^l_i = \text{round}(p^l_i)$, while a stochastic sampling from Bernoulli distribution is possible as well.

Note that the whole model is differentiable except for $f_{binarize}$ because the $\arg\max$ operation is a hard-decision process and the gates have discrete values of 0 and 1. Thus, errors cannot be back-propagated through gradient descent. A common method for optimizing models involving discrete variables is REINFORCE (Williams, 1992). However, the REINFORCE algorithms suffer from model instability, and hard training (Maddison et al., 2016). We instead use a Gumbel-Softmax distribution (Gumbel, 1948; Jang et al., 2016) to approximate Equation 9 as follow:

$$z^l_i = \frac{\exp((\log(p^l_{i,1}) + \epsilon_1)/\tau)}{\sum_{t=0}^1 \exp((\log(p^l_{i,t}) + \epsilon_t)/\tau))}, \tag{10}$$

where $\epsilon_t$ is a random sample from Gumbel$(0,1)$[1] and $\tau$ is the temperature coefficient. When $\tau \to 0$, Equation 10 approaches the $\arg\max$ operation. During training, We use the gradients of Gumbel-Softmax as the surrogate gradients for error back-propagation. At test time, the surrogate is not necessary and the generated gates are binary as Equation 9.

## 3.3 Pooling

With $L$ layers of our DP-GCN, we obtain the hidden representations of all tokens at each layer. The role of the pooling module is to aggregate such vectors to generate the most informative features as relational representation. Specifically, a linear combination is deployed to integrate representations from different layers, allowing rich local and non-local information to be captured:

$$\mathbf{h}^{comb}_i = \mathbf{W}_{comb}[\mathbf{h}^1_i; \cdots; \mathbf{h}^L_i] + \mathbf{b}_{comb}, \tag{11}$$

where $\mathbf{h}^{comb}_i$ is the combined feature vector of token $i$, $\mathbf{W}_{comb} \in \mathbb{R}^{d\times Ld)}$ is a weight matrix and $\mathbf{b}_{comb}$ is a bias vector. A max-pooling operation (denoted as $\mathcal{F}$) is further applied to capture the most important semantic features for the entire sentence: $\mathbf{h}_{sent} = \mathcal{F}(\mathbf{h}^{comb}_{1:n})$. Similarly, we can obtain the subject

[1]Gumbel$(0,1) = -\log(-\log(\text{Uniform}(0,1)))$

representation $\mathbf{h}_{subj} = \mathcal{F}(\mathbf{h}^{comb}_{s_1:s_2})$ and object representation $\mathbf{h}_{obj} = \mathcal{F}(\mathbf{h}^{comb}_{o_1:o_2})$, where $s_1$, $s_2$ are the starting and ending indices of subject, respectively, $o_1$, $o_2$ denote the boundary indices of object. Following recent works (Zhang et al., 2018; Guo et al., 2019), we obtain the relational representation used for classification by concatenating the sentence and entity representations:

$$\mathbf{r} = [\mathbf{h}_{sent}; \mathbf{h}_{subj}; \mathbf{h}_{obj}]. \tag{12}$$

### 3.4 Rethinking Mechanism

Under the above framework, $\mathbf{m}^l$, $\mathbf{s}^l$ and $\mathbf{o}^l$ in Equation 7 play an indispensable role in the dynamic pruning process. Since these features determine the selection module's perception of the target entity pair and the entire sentence. With the progress of entity and sentence understanding, the selection module would produce more precise pruning results. Motivated by this intuition, in this work, we develop the rethinking mechanism to pay close attention to the most important nodes with the consideration of learned information. To be specific, as shown in Figure 2, we treat the output of pooling module as the high-level features, and use these features to adjust the gate values of the selection module by introducing feedback connections to each DP-GCN layer, such rethinking process can be performed repeatedly. In other words, we reuse $\mathbf{h}_{sent}, \mathbf{h}_{subj}, \mathbf{h}_{obj}$ of the previous rethinking step as $\mathbf{m}^l, \mathbf{s}^l, \mathbf{o}^l$ at each layer's selection module of the current step. In this way, the network is endowed with the ability to adaptively refine the pruning operation for better target-specific semantic understanding. As for the first step that $\mathbf{h}_{sent}, \mathbf{h}_{subj}, \mathbf{h}_{obj}$ cannot be provided, we set all gates as open ($z = 1$) without making node selection.

### 3.5 Model Training

The produced relational representation $\mathbf{r}$ from the last rethinking step is fed into a feed-forward neural network (FFNN), followed by a Softmax normalization layer to yield a probability distribution over relational decision space:

$$p(\hat{\mathbf{y}}|\mathbf{r}) = \text{Softmax}(\text{FFNN}(\mathbf{r})), \tag{13}$$

where $\hat{\mathbf{y}}$ is the predicted relational distribution. During the training, we optimize the parameters of the entire network to minimize the cross-entropy loss:

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_i \text{log} p(\hat{\mathbf{y}}_i|\mathbf{r}_i), \tag{14}$$

where $\mathbf{y}_i$ is the one-hot vector represented ground truth of the $i$-th instance, and $N$ denotes the number of training instances.

## 4 Experiments

### 4.1 Dataset and Metric

We conduct experiments on two relation extraction datasets: (1) **TACRED** (Zhang et al., 2017): It is the currently largest benchmark dataset for supervised relation extraction, which contains 41 relations and a specially *no-relation* class indicating that the relation expressed in the sentence is not among the 41 types. TACRED is partitioned into training (68124 samples), dev (22631 samples) and test (15509 samples) sets, we tune the hyper-parameters according to results on the dev set. Mentions in TACRED are typed to avoid overfitting on specific entities and provide entity type information, in which subjects fall into 2 categories, and objects are categorized into 16 types. We report micro-averaged Precision, Recall and F$_1$ scores on this dataset as is conventional. (2) **SemEval** (Hendrickx et al., 2009) : The SemEval (i.e., SemEval 2010 task 8) dataset contains 9 directed relations and a *no-relation* class. It is smaller and simpler than TACRED with 8000 training samples and 2717 test samples. We use this dataset to evaluate the generalization ability of our proposed model. On SemEval, we follow the convention and report the macro-averaged F$_1$ scores. For fair comparisons, we report the averaged test results $\pm$ one standard deviation over 5 randomly initialized runs.

| System | Precision | Recall | $F_1$ |
|---|---|---|---|
| SDP-LSTM (Xu et al., 2015b) | 66.3 | 52.7 | 58.7 |
| LR (Zhang et al., 2017) | **73.5** | 49.9 | 59.4 |
| PA-LSTM (Zhang et al., 2017) | 65.7 | 64.5 | 65.1 |
| C-GCN (Zhang et al., 2018) | 69.9 | 63.3 | 66.4 |
| SA-LSTM (Yu et al., 2019) | 69.0 | 66.2 | 67.6 |
| KnwlSelf (Li et al., 2019) | 67.1 | **68.4** | 67.8 |
| ERNIE (Zhang et al., 2019) | 70.0 | 66.1 | 68.0 |
| AGGCN (Guo et al., 2019) | 73.1 $(71.6 \pm 0.4)^{\dagger}$ | 64.2 $(63.6 \pm 0.3)^{\dagger}$ | 68.2 $(67.4 \pm 0.3)^{\dagger}$ |
| DP-GCN | $72.2 \pm 0.3$ | $66.5 \pm 0.2^{\star}$ | **69.2** $\pm 0.2^{\star}$ |

Table 1: Micro-averaged precision, recall and $F_1$ score on the TACRED test set. The best performance is in bold for each metric. † marks results produced from re-running the official source code, which are consistent with the numbers reported by other researchers [2]. $\star$ marks statistically significant improvements over AGGCN with $p < 0.01$ under a bootstrap test.

### 4.2 Implementation Details

The model is trained with SGD optimizer with the initial learning rate of $0.7$ and the weight decay of $0.9$. Following previous studies (Zhang et al., 2018; Guo et al., 2019), we exploit 300-dimensional Glove (Pennington et al., 2014) vectors for the word embeddings, and generate dependency parse trees with Stanford CoreNLP (Manning et al., 2014). We choose the temperature $\tau$ in Gumbel-Softmax from the set $\{0.1, 0.3, 0.5, 0.7\}$, the rethinking times from $\{1, 2, 3, 4, 5\}$. We use 3 DP-GCN layers in our experiments, and to fully capture the dependency information, in the first layer, we directly feed the original dependency tree to the GCN module without any pruning. To avoid deleting all nodes in the graph, we set the gates of the target entity nodes as open during training and test time. The hidden state size of BiLSTM and DP-GCN are both set to 300. To ease overfitting, we apply dropout on the word embeddings and each DP-GCN layer with rate $0.5$.

### 4.3 Comparison Models

In experiments, we compare our DP-GCN model with two groups of methods:

**Dependency-based models.** (1) SDP-LSTM (Xu et al., 2015b): it applies a neural sequence model on the shortest dependency path between the subject and object entities. (2) LR (Zhang et al., 2017): a logistic regression classifier that combines dependency-based features with other lexical features. (3) C-GCN (Zhang et al., 2018): a contextualized GCN over the dependency tree where the input vectors are obtained using bi-directional LSTM network, a path-centric pruning is also introduced to remove irrelevant content. (4) AGGCN (Guo et al., 2019): an attention guided graph convolutional network, which transforms the dependency tree into a fully connected graph by multi-head self-attention, and achieves the recent state-of-the-art performance on the TACRED dataset.

**Neural sequence models.** (1) PA-LSTM (Zhang et al., 2017): it employs a position-aware attention mechanism to summarize the LSTM outputs, and outperforms several strong baselines. (2) SA-LSTM (Yu et al., 2019): it adopts a segment attention mechanism on top of the LSTM, and is capable of learning relational expressions. (3) ERNIE (Zhang et al., 2019): it is a pre-trained language model with rich knowledge information, and outperforms BERT in this task. (4) KnwlSelf (Li et al., 2019): a knowledge-attention encoder that incorporates prior knowledge from external lexical resources such as FrameNet into a self-attention network.

### 4.4 Main Results

Table 1 summarizes the experimental results on the TACRED test set. Generally speaking, our proposed model significantly outperforms competing baselines and achieves the best $F_1$ score. Over AGGCN,

[2]https://github.com/Cartus/AGGCN/issues/12

| Model | $F_1$ |
|---|---|
| PA-LSTM (Zhang et al., 2017) | 82.7 |
| C-GCN (Zhang et al., 2018) | 84.8 |
| KnwlSelf (Li et al., 2019) | 84.3 |
| AGGCN (Guo et al., 2019) | $85.4 \pm 0.3$ |
| DP-GCN | **86.4** $\pm 0.3^{\star}$ |

Table 2: Macro-averaged $F_1$ score on SemEval.

| Model | Dev $F_1$ |
|---|---|
| Best DP-GCN | 69.0 |
| – selection module | 67.2 |
| – rethinking mechanism | 68.2 |
| – dependency tree structure | 67.5 |
| – $f_{binarize}$ function | 68.4 |

Table 3: An ablation study on the TACRED dev set.

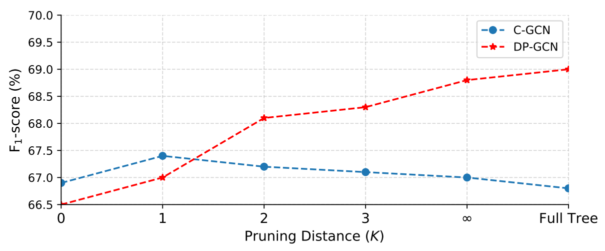

Figure 3: The impact of pruning strategies.

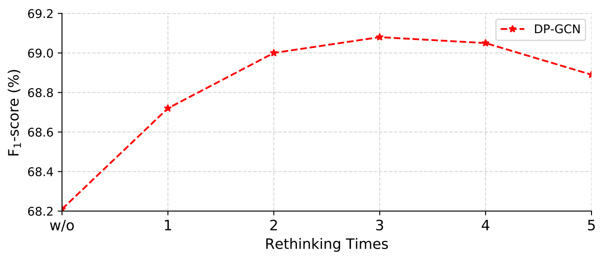

Figure 4: The impact of rethinking times.

DP-GCN achieves an absolute improvement of 1.8% in $F_1$ score, the gain mainly comes from improved recall and we hypothesize that this is because DP-GCN introduces the entity information into graph modeling process to control the flow of information, and therefore retains more discriminative features related to the target entity pair compared to AGGCN which constructs a target-irrelevant dense graph. Meanwhile, DP-GCN improves upon C-GCN in both precision and recall, which verifies the superiority of our proposed dynamic pruning strategy against the hand-crafted pruning rule. We also observe that DP-GCN's performance exceeds existing neural sequence models, especially in terms of accuracy. This shows that the syntactic information obtained from dependency parsing is effective in capturing long-range syntactic relations between entities.

To further demonstrate the advantage of our model, we also evaluate DP-GCN on the SemEval dataset (Table 2) under the same settings as C-GCN (Zhang et al., 2018) and AGGCN (Guo et al., 2019). Our DP-GCN model consistently outperforms baseline models, exhibiting great generalizability.

### 4.5 Ablation Study

To demonstrate the effectiveness of each component, we discard one particular component at a time to understand its impact on the performance. From these ablations, we find that: (1) The entire selection module contributes about 1.8% $F_1$ score. (2) When we remove the rethinking mechanism and compute $\mathbf{s}^l = \mathcal{F}(\mathbf{h}^l_{s_1:s_2}), \mathbf{o}^l = \mathcal{F}(\mathbf{h}^l_{o_1:o_2}), \mathbf{m}^l = \mathcal{F}(\mathbf{h}^l_{1:n})$, where $\mathcal{F}$ denotes the max-pooling operation, the score drops by 0.8%, which indicates that rethinking is efficient in leveraging the high-level learned semantic to guide and refine the pruning process. (3) Removing the dependency structure (i.e., directly applying the GCN module over $\tilde{\mathbf{A}}^l$) hurts the result by 1.5% $F_1$ score. This implies that the syntactic information introduced by dependency trees is important and needed. (4) By binarizing the gate probability, we can filter irrelevant information more effectively, which is consistent with the conclusion in previous works using hard selection (Lei et al., 2019; Xue et al., 2020).

### 4.6 Analysis on Pruning Strategies

In order to better verify the pruning ability of DP-GCN, we preprocess the input sentence of C-GCN and DP-GCN with the same pruning rule which only keeps the tokens that are up to distance $K$ away from the SDP in the LCA subtree, and also include results when the full tree is used. $K = 0$ corresponds to pruning the tree down to the SDP, and $K = \infty$ retains the entire LCA subtree. As illustrated in Figure 3, the performance of C-GCN on the TACRED dev set peaks when $K = 1$, outperforming its

|  | Example | Predicted relation | True relation |
|---|---|---|---|
| C-GCN($K = 1$) | He said that with the sales **of SUBJ-ORG and the Asian unit to OBJ-ORG,** the company generated 50.7 billion dollars | *no-relation* | *org:parents* |
| DP-GCN | He said that with **the sales of SUBJ-ORG** and the Asian **unit to OBJ-ORG**, the company generated 50.7 billion dollars | *org:parents* |  |
| C-GCN($K = 1$) | Survivors include **SUBJ-PER wife, OBJ-PER;** three **sons**, Jeff, James and Harris**;** a **daughter,** Leslie; **and mother**, Sally. | *per:spouse* | *per:spouse* |
| DP-GCN | Survivors include **SUBJ-PER wife, OBJ-PER;** three sons, Jeff, James **and** Harris; a daughter, Leslie; and mother, Sally. | *per:children* |  |

Table 4: Case study on TACRED. Bold texts are focused tokens selected by C-GCN($K = 1$) and DP-GCN (the last layer) respectively. The third column for each example is the predicted result of the corresponding model and the fourth column is the gold standard.

full tree-based counterpart. This confirms the hypothesis in previous studies (Xu et al., 2015a; Zhang et al., 2017) that not all tokens in the dependency tree are needed to express the target relation, and removing target-irrelevant tokens could improve the performance. However, for our DP-GCN model, taking pruned trees as input is not effective, and pruning more aggressively could lead to worse results. These observations demonstrate that our model has learned to dynamically prune the dependency tree for the target entity pair, thus any pre-defined pruning rules may mistakenly remove useful information and affect the performance of DP-GCN.

### 4.7 Analysis on Rethinking Times

In this subsection, we study the performance of our proposed model with different times of rethinking. The detailed results on the TACRED dev set are shown in Figure 4. We can find that, with rethinking times increasing from 0 (w/o) to 3, the $F_1$ score increases from 68.2 to 69.1. This verifies that the rethinking mechanism can enhance the pruning ability by allowing the bottom layers to receive richer top-down information. When the number of rethinking times surpasses 3, we observe the performance declines instead to some extent. One possible reason is that, with the increase of rethinking times, the model may pay much attention to the target entities and ignore other crucial relational features. Besides, it is obvious that rethinking repeatedly will inevitably increase the runtime (the time of each training batch increases from 0.08s to 0.17s when the number of rethinking times increases from 0 to 3). In order to trade off the time cost and the final performance, we choose to rethink twice in our experiments.

### 4.8 Case Study

To gain insights into the behavior of our model, we conduct a case study as shown in Table 4. As demonstrated by the first example, our DP-GCN model successfully identifies target-relevant clues “the sales of” while the hard pruning strategy focuses on some unimportant tokens. As a result, C-GCN is not able to capture the interactions between removed tokens and entities, since these tokens are not in the resulting structure. Hence, it is not surprising that C-GCN wrongly marks this instance as *no-relation*. In the second example, C-GCN predicts the relation to be *per:children* rather than *per:spouse*. We hypothesize the reason is that the pruned tree includes some noisy target-irrelevant tokens (i.e., “sons” and “daughter”) which confuse the classification. So it is difficult for C-GCN to distinguish between relation *per:children* and *per:spouse*. Thanks to the dynamic selection module, DP-GCN successfully identifies critical tokens that provide sufficient information to extract the relation between two entities. From these examples, we can observe that the proposed model is capable of learning to prune the dependency tree in an entity-specific manner to perform relation extraction.

## 5 Conclusion

In this paper, we propose a novel model that learns to prune dependency trees for relation extraction in an end-to-end manner. By incorporating a selection model into each GCN layer, our model is capable of filtering target-irrelevant information without relying on any pre-defined rules. We further introduce a rethinking mechanism to guide and adjust the pruning operation by feeding back the high-level semantic repeatedly. Experiments on two public datasets show that our proposed model outperforms several strong baselines and achieves state-of-the-art performance. In the future, we will conduct research on how to design a more sophisticated pruning method to better leverage the dependency structure by focusing on the crucial content more precisely.

## 6 Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This research is supported by the National Key Research and Development Program of China (grant No.2016YFB0801003) and the Strategic Priority Research Program of Chinese Academy of Sciences (grant No.XDC02040400).

## References

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of EMNLP*.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of ACL*.

Duy-Cat Can, Hoang-Quynh Le, Quang-Thuy Ha, and Nigel Collier. 2019. A richer-but-smarter shortest dependency path with attentive augmentation for relation extraction. In *Proceedings of NAACL-HLT*.

Zihang Dai, Lei Li, and Wei Xu. 2016. Cfo: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of ACL*.

Javid Ebrahimi and Dejing Dou. 2015. Chain based rnn for relation classification. In *Proceedings of ACL*.

Tao Gui, Ruotian Ma, Qi Zhang, Lujun Zhao, and Xuanjing Huang. 2019. Cnn-based chinese ner with lexicon rethinking. In *Proceedings of IJCAI*.

Emil Julius Gumbel. 1948. *Statistical theory of extreme values and some practical applications: a series of lectures*. US Government Printing Office.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of ACL*.

Zhengqiu He, Wenliang Chen, Zhenghua Li, Meishan Zhang, Wei Zhang, and Min Zhang. 2018. See: Syntax-aware entity embedding for neural relation extraction. In *Proceedings of AAAI*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid O 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. *SEW-2009 Semantic Evaluations: Recent Achievements and Future Directions*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. In *Proceedings of ICLR*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Zeyang Lei, Yujiu Yang, Min Yang, Wei Zhao, Jun Guo, and Yi Liu. 2019. A human-like semantic cognition network for aspect-level sentiment classification. In *Proceedings of AAAI*.

Xin Li, Zequn Jie, Jiashi Feng, Changsong Liu, and Shuicheng Yan. 2018. Learning with rethinking: Recurrently improving convolutional neural networks through feedback. *Pattern Recognition*.

Pengfei Li, Kezhi Mao, Xuefeng Yang, and Qi Li. 2019. Improving relation extraction with knowledge-attention. In *Proceedings of EMNLP*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, and Ming Zhou. 2015. A dependency-based neural network for relation classification. In *Proceedings of ACL*.

Chris J Maddison, Andriy Mnih, and Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of ICLR*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of ACL*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Kai Sun, Richong Zhang, Yongyi Mao, Samuel Mensah, and Xudong Liu. 2020. Relation extraction with convolutional network over learnable syntax-transport graph. In *Proceedings of AAAI*.

Van-Hien Tran, Van-Thuy Phi, Hiroyuki Shindo, and Yuji Matsumoto. 2019. Relation classification using segment-level attention-based cnn and dependency-based rnn. In *Proceedings of NAACL-HLT*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of EMNLP*.

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 EMNLP*.

Lanqing Xue, Xiaopeng Li, and Nevin L Zhang. 2020. Not all attention is needed: Gated attention network for sequence data. In *Proceedings of AAAI*.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Proceedings of AAAI*.

Bowen Yu, Zhenyu Zhang, Tingwen Liu, Bin Wang, Sujian Li, and Quangang Li. 2019. Beyond word attention: using segment attention in neural relation extraction. In *Proceedings of IJCAI*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*.

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 EMNLP*.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of ACL*.