Attention Guided Graph Convolutional Networks for Relation Extraction

Zhijiang Guo[*] , **Yan Zhang**[*] and **Wei Lu**
StatNLP Research Group
Singapore University of Technology and Design
{zhijiang_guo,yan_zhang}@mymail.sutd.edu.sg, luwei@sutd.edu.sg

Abstract

Dependency trees convey rich structural information that is proven useful for extracting relations among entities in text. However, how to effectively make use of relevant information while ignoring irrelevant information from the dependency trees remains a challenging research question. Existing approaches employing rule based hard-pruning strategies for selecting relevant partial dependency structures may not always yield optimal results. In this work, we propose Attention Guided Graph Convolutional Networks (AGGCNs), a novel model which directly takes full dependency trees as inputs. Our model can be understood as a soft-pruning approach that automatically learns how to selectively attend to the relevant sub-structures useful for the relation extraction task. Extensive results on various tasks including cross-sentence n-ary relation extraction and large-scale sentence-level relation extraction show that our model is able to better leverage the structural information of the full dependency trees, giving significantly better results than previous approaches.

1 Introduction

Relation extraction aims to detect relations among entities in the text. It plays a significant role in a variety of natural language processing applications including biomedical knowledge discovery (Quirk and Poon, 2017), knowledge base population (Zhang et al., 2017) and question answering (Yu et al., 2017). Figure 1 shows an example about expressing a relation sensitivity among three entities *L858E*, *EGFR* and *gefitinib* in two sentences.

Most existing relation extraction models can be categorized into two classes: sequence-based and dependency-based. Sequence-based models operate only on the word sequences (Zeng et al.,

2014; Wang et al., 2016), whereas dependency-based models incorporate dependency trees into the models (Bunescu and Mooney, 2005; Peng et al., 2017). Compared to sequence-based models, dependency-based models are able to capture non-local syntactic relations that are obscure from the surface form alone (Zhang et al., 2018). Various pruning strategies are also proposed to distill the dependency information in order to further improve the performance. Xu et al. (2015b,c) apply neural networks only on the shortest dependency path between the entities in the full tree. Miwa and Bansal (2016) reduce the full tree to the subtree below the lowest common ancestor (LCA) of the entities. Zhang et al. (2018) apply graph convolutional networks (GCNs) (Kipf and Welling, 2017) 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, rule-based pruning strategies might eliminate some important information in the full tree. Figure 1 shows an example in cross-sentence n-ary relation extraction that the key tokens *partial response* would be excluded if the model only takes the pruned tree into consideration.[1] Ideally, the model should be able to learn how to maintain a balance between including and excluding information in the full tree. In this paper, we propose the novel Attention Guided Graph Convolutional Networks (AGGCNs), which operate directly on the full tree. Intuitively, we develop a "soft pruning" strategy that transforms the original dependency tree into a fully connected edge-weighted graph. These weights can be viewed as the strength of relatedness between nodes, which can be learned in an end-to-end fashion by using self-attention mechanism (Vaswani et al., 2017).

In order to encode a large fully connected graph,

---

[*] Equally Contributed.

[1]Case study and attention visualization of this example are provided in the supplementary material.
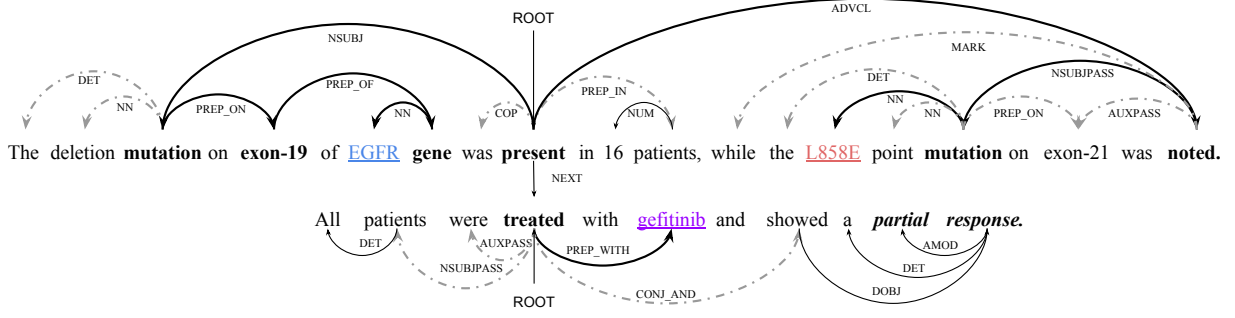
Figure 1: An example dependency tree for two sentences expressing a relation (sensitivity) among three entities. The shortest dependency path between these entities is highlighted in bold (edges and tokens). The root node of the LCA subtree of entities is *present*. The dotted edges indicate tokens K=1 away from the subtree. Note that tokens *partial response* off these paths (shortest dependency path, LCA subtree, pruned tree when K=1).

we next introduce dense connections (Huang et al., 2017) to the GCN model following (Guo et al., 2019). For GCNs, L layers will be needed in order to capture neighborhood information that is L hops away. A shallow GCN model may not be able to capture non-local interactions of large graphs. Interestingly, while deeper GCNs can capture richer neighborhood information of a graph, empirically it has been observed that the best performance is achieved with a 2-layer model (Xu et al., 2018). With the help of dense connections, we are able to train the AGGCN model with a large depth, allowing rich local and non-local dependency information to be captured.

Experiments show that our model is able to achieve better performance for various tasks. For the cross-sentence relation extraction task, our model surpasses the current state-of-the-art models on multi-class ternary and binary relation extraction by 8% and 6% in terms of accuracy respectively. For the large-scale sentence-level extraction task (TACRED dataset), our model is also consistently better than others, showing the effectiveness of the model on a large training set. Our code is available at http://www.statnlp.org/research/information-extraction[2]

Our contributions are summarized as follows:

- We propose the novel AGGCNs that learn a "soft pruning" strategy in an end-to-end fashion, which learns how to select and discard information. Combining with dense connections, our AGGCN model is able to learn a better graph representation.
- Our model achieves new state-of-the-art re-

sults without additional computational overhead when compared with previous GCNs.[3] Unlike tree-structured models (e.g., Tree-LSTM (Tai et al., 2015)), it can be efficiently applied over dependency trees in parallel.

2 Attention Guided GCNs

In this section, we will present the basic components used for constructing our AGGCN model.

2.1 GCNs

GCNs are neural networks that operate directly on graph structures (Kipf and Welling, 2017). Here we mathematically illustrate how multi-layer GCNs work on a graph. Given a graph with n nodes, we can represent the graph with an $n \times n$ adjacency matrix $\mathbf{A}$. Marcheggiani and Titov (2017) extend GCNs for encoding dependency trees by incorporating directionality of edges into the model. They add a self-loop for each node in the tree. Opposite direction of a dependency arc is also included, which means $\mathbf{A}_{ij} = 1$ and $\mathbf{A}_{ji} = 1$ if there is an edge going from node i to node j, otherwise $\mathbf{A}_{ij} = 0$ and $\mathbf{A}_{ji} = 0$. The convolution computation for node i at the l-th layer, which takes the input feature representation $\mathbf{h}^{(l-1)}$ as input and outputs the induced representation $\mathbf{h}_i^{(l)}$, can be defined as:

$$\mathbf{h}_i^{(l)} = \rho\Big(\sum_{j=1}^{n} \mathbf{A}_{ij} \mathbf{W}^{(l)} \mathbf{h}_j^{(l-1)} + \mathbf{b}^{(l)} \Big) \quad (1)$$

where $\mathbf{W}^{(l)}$ is the weight matrix, $\mathbf{b}^{(l)}$ is the bias vector, and ρ is an activation function (e.g.,

---

[2] Implementation is based on Pytorch (Paszke et al., 2017).

[3] The size of the adjacency matrix representing the fully connected graph is the same as the one of the original tree.
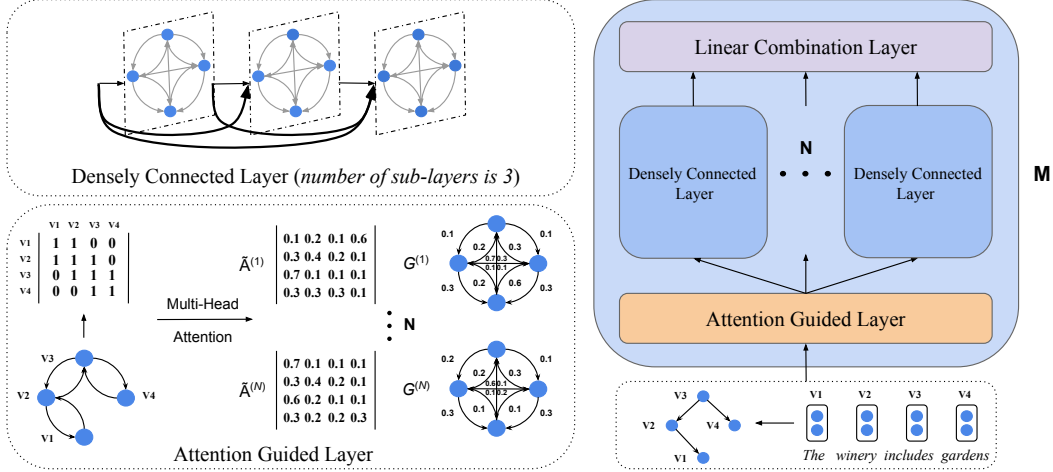
Figure 2: The AGGCN model is shown with an example sentence and its dependency tree. It is composed of M identical blocks and each block has three types of layers as shown on the right. Every block takes node embeddings and adjacency matrix that represents the graph as inputs. Then N attention guided adjacency matrices are constructed by using multi-head attention as shown at bottom left. The original dependency tree is transformed into N different fully connected edge-weighted graphs (self-loops are omitted for simplification). Numbers near the edges represent the weights in the matrix. Resulting matrices are fed into N separate densely connected layers, generating new representations. Top left shows an example of the densely connected layer, where the number (L) of sub-layers is 3 (L is a hyper-parameter). Each sub-layer concatenates all preceding outputs as the input. Eventually, a linear combination is applied to combine outputs of N densely connected layers into hidden representations.

RELU). $\mathbf{h}_i^{(0)}$ is the initial input $\mathbf{x}_i$, where $\mathbf{x}_i \in \mathbb{R}^d$ and d is the input feature dimension.

2.2 Attention Guided Layer

The AGGCN model is composed of M identical blocks as shown in Figure 2. Each block consists of three types of layers: attention guided layer, densely connected layer and linear combination layer. We first introduce the attention guided layer of the AGGCN model.

As we discuss in Section 1, most existing pruning strategies are predefined. They prune the full tree into a subtree, based on which the adjacency matrix is constructed. In fact, such strategies can also be viewed as a form of hard attention (Xu et al., 2015a), where edges that connect nodes not on the resulting subtree will be directly assigned zero weights (not attended). Such strategies might eliminate relevant information from the original dependency tree. Instead of using rule-based pruning, we develop a "soft pruning" strategy in the attention guided layer, which assigns weights to all edges. These weights can be learned by the model in an end-to-end fashion.

In the attention guided layer, we transform the original dependency tree into a fully connected edge-weighted graph by constructing an attention guided adjacency matrix $\tilde{\mathbf{A}}$. Each $\tilde{\mathbf{A}}$ corresponds to a certain fully connected graph and each entry

$\tilde{\mathbf{A}}_{ij}$ is the weight of the edge going from node i to node j. As shown in Figure 2, $\tilde{\mathbf{A}}^{(1)}$ represents a fully connected graph $G^{(1)}$. $\tilde{\mathbf{A}}$ can be constructed by using self-attention mechanism (Cheng et al., 2016), which is an attention mechanism (Bahdanau et al., 2015) that captures the interactions between two arbitrary positions of a single sequence. Once we get $\tilde{\mathbf{A}}$, we can use it as the input for the computation of the later graph convolutional layer. Note that the size of $\tilde{\mathbf{A}}$ is the same as the original adjacency matrix $\mathbf{A}$ ($n \times n$). Therefore, no additional computational overhead is involved. The key idea behind the attention guided layer is to use attention for inducing relations between nodes, especially for those connected by indirect, multi-hop paths. These soft relations can be captured by differentiable functions in the model.

Here we compute $\tilde{\mathbf{A}}$ by using multi-head attention (Vaswani et al., 2017), which allows the model to jointly attend to information from different representation subspaces. The calculation involves a query and a set of key-value pairs. The output is computed as a weighted sum of the values, where the weight is computed by a function of the query with the corresponding key.

$$\tilde{\mathbf{A}}^{(\mathbf{t})} = softmax(\frac{Q\mathbf{W}_i^Q \times (K\mathbf{W}_i^K)^T}{\sqrt{d}})V \quad (2)$$

where Q and K are both equal to the collective representation $\mathbf{h}^{(l-1)}$ at layer $l-1$ of the AG-GCN model. The projections are parameter matrices $\mathbf{W}_i^Q \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_i^K \in \mathbb{R}^{d \times d}$. $\tilde{\mathbf{A}}^{(\mathbf{t})}$ is the t-th attention guided adjacency matrix corresponding to the t-th head. Up to N matrices are constructed, where N is a hyper-parameter.

Figure 2 shows an example that the original adjacency matrix is transformed into multiple attention guided adjacency matrices. Accordingly, the input dependency tree is converted into multiple fully connected edge-weighted graphs. In practice, we treat the original adjacency matrix as an initialization so that the dependency information can be captured in the node representations for later attention calculation. The attention guided layer is included starting from the second block.

2.3 Densely Connected Layer

Unlike previous pruning strategies, which lead to a resulting structure that is smaller than the original structure, our attention guided layer outputs a larger fully connected graph. Following (Guo et al., 2019), we introduce dense connections (Huang et al., 2017) into the AGGCN model in order to capture more structural information on large graphs. With the help of dense connections, we are able to train a deeper model, allowing rich local and non-local information to be captured for learning a better graph representation.

Dense connectivity is shown in Figure 2. Direct connections are introduced from any layer to all its preceding layers. Mathematically, we first define $\mathbf{g}_j^{(l)}$ as the concatenation of the initial node representation and the node representations produced in layers $1, \cdots, l-1$:

$$\mathbf{g}_j^{(l)} = [\mathbf{x}_j; \mathbf{h}_j^{(1)}; ...; \mathbf{h}_j^{(l-1)}]. \quad (3)$$

In practice, each densely connected layer has L sub-layers. The dimensions of these sub-layers d_{hidden} are decided by L and the input feature dimension d. In AGGCNs, we use $d_{hidden} = d/L$. For example, if the densely connected layer has 3 sub-layers and the input dimension is 300, the hidden dimension of each sub-layer will be $d_{hidden} = d/L = 300/3 = 100$. Then we concatenate the output of each sub-layer to form the new representation. Therefore, the output dimension is 300 (3×100). Different from the GCN model whose hidden dimension is larger than or equal to the input dimension, the AGGCN model shrinks the hidden dimension as the number of layers increases in

order to improves the parameter efficiency similar to DenseNets (Huang et al., 2017).

Since we have N different attention guided adjacency matrices, N separate densely connected layers are required. Accordingly, we modify the computation of each layer as follows (for the t-th matrix $\tilde{\mathbf{A}}^{(\mathbf{t})}$):

$$\mathbf{h}_{t_i}^{(l)} = \rho\Big(\sum_{j=1}^n \tilde{\mathbf{A}}_{ij}^{(t)} \mathbf{W}_t^{(l)} \mathbf{g}_j^{(l)} + \mathbf{b}_t^{(l)} \Big) \quad (4)$$

where $t = 1, ..., N$ and t selects the weight matrix and bias term associated with the attention guided adjacency matrix $\tilde{\mathbf{A}}^{(\mathbf{t})}$. The column dimension of the weight matrix increases by d_{hidden} per sub-layer, i.e., $\mathbf{W}_t^{(l)} \in \mathbb{R}^{d_{hidden} \times d^{(l)}}$, where $d^{(l)} = d + d_{hidden} \times (l-1)$.

2.4 Linear Combination Layer

The AGGCN model includes a linear combination layer to integrate representations from N different densely connected layers. Formally, the output of the linear combination layer is defined as:

$$\mathbf{h}_{comb} = \mathbf{W}_{comb} \mathbf{h}_{out} + \mathbf{b}_{comb} \quad (5)$$

where $\mathbf{h}_{out}$ is the output by concatenating outputs from N separate densely connected layers, i.e., $\mathbf{h}_{out} = [\mathbf{h}^{(1)}; ...; \mathbf{h}^{(N)}] \in \mathbb{R}^{d \times N}$. $\mathbf{W}_{comb} \in \mathbb{R}^{(d \times N) \times d}$ is a weight matrix and $\mathbf{b}_{comb}$ is a bias vector for the linear transformation.

2.5 AGGCNs for Relation Extraction

After applying the AGGCN model over the dependency tree, we obtain hidden representations of all tokens. Given these representations, the goal of relation extraction is to predict a relation among entities. Following (Zhang et al., 2018), we concatenate the sentence representation and entity representations to get the final representation for classification. First we need to obtain the sentence representation h_{sent}. It can be computed as:

$$h_{sent} = f(\mathbf{h}_{\mathbf{mask}}) = f(\text{AGGCN}(\mathbf{x})) \quad (6)$$

where $\mathbf{h}_{\mathbf{mask}}$ represents the masked collective hidden representations. Masked here means we only select representations of tokens that are not entity tokens in the sentence. $f : \mathbb{R}^{d \times n} \to \mathbb{R}^{d \times 1}$ is a max pooling function that maps from n output vectors to 1 sentence vector. Similarly, we can obtain the entity representations. For the i-th entity, its representation h_{e_i} can be computed as:

$$h_{e_i} = f(\mathbf{h}_{\mathbf{e_i}}) \quad (7)$$

where $\mathbf{h_{e_i}}$ indicates the hidden representation corresponding to the i-th entity.[4] Entity representations will be concatenated with sentence representation to form a new representation. Following (Zhang et al., 2018), we apply a feed-forward neural network (FFNN) over the concatenated representations inspired by relational reasoning works (Santoro et al., 2017; Lee et al., 2017):

$$h_{final} = \text{FFNN}([h_{sent}; h_{e_1}; ...h_{e_i}]) \qquad (8)$$

where h_{final} will be taken as inputs to a logistic regression classifier to make a prediction.

3 Experiments

3.1 Data

We evaluate the performance of our model on two tasks, namely, cross-sentence n-ary relation extraction and sentence-level relation extraction.

For the cross-sentence n-ary relation extraction task, we use the dataset introduced in (Peng et al., 2017), which contains 6,987 ternary relation instances and 6,087 binary relation instances extracted from PubMed.[5] Most instances contain multiple sentences and each instance is assigned with one of the five labels, including "resistance or nonresponse", "sensitivity", "response", "resistance" and "none". We consider two specific tasks for evaluation, i,e., binary-class n-ary relation extraction and multi-class n-ary relation extraction. For binary-class n-ary relation extraction, we follow (Peng et al., 2017) to binarize multi-class labels by grouping the four relation classes as "yes" and treating "none" as "no".

For the sentence-level relation extraction task, we follow the experimental settings in (Zhang et al., 2018) to evaluate our model on the TACRED dataset (Zhang et al., 2017) and Semeval-10 Task 8 (Hendrickx et al., 2010). With over 106K instances, the TACRED dataset introduces 41 relation types and a special "no relation" type to describe the relations between the mention pairs in instances. Subject mentions are categorized into "person" and "organization", while object mentions are categorized into 16 fine-grained types, including "date", "location", etc. Semeval-10 Task 8 is a public dataset, which contains 10,717 instances with 9 relations and a special "other" class.

3.2 Setup

We tune the hyper-parameters according to results on the development sets. For the cross-sentence n-ary relation extraction task, we use the same data split used in (Song et al., 2018b)[5], while for the sentence-level relation extraction task, we use the same development set from (Zhang et al., 2018)[6].

We choose the number of heads N for attention guided layer from $\{1, 2, 3, 4\}$, the block number M from $\{1, 2, 3\}$, the number of sublayers L in each densely connected layer from $\{2, 3, 4, 5, 6\}$. Through preliminary experiments on the development sets, we find that the combinations (N=2, M=2, L=5, d_{hidden}=340) and (N=3, M=2, L=5, d_{hidden}=300) give the best results on cross-sentence n-ary relation extraction and sentence-level relation extraction, respectively. GloVe (Pennington et al., 2014)[7] vectors are used as the initialization for word embeddings.

Models are evaluated using the same metrics as previous work (Song et al., 2018b; Zhang et al., 2018). We report the test accuracy averaged over five cross validation folds (Song et al., 2018b) for the cross-sentence n-ary relation extraction task. For the sentence-level relation extraction task, we report the micro-averaged F1 scores for the TACRED dataset and the macro-averaged F1 scores for the SemEval dataset (Zhang et al., 2018).

3.3 Results on Cross-Sentence n-ary Relation Extraction

For cross-sentence n-ary relation extraction task, we consider three kinds of models as baselines: 1) a feature-based classifier (Quirk and Poon, 2017) based on shortest dependency paths between all entity pairs, 2) Graph-structured LSTM methods, including Graph LSTM (Peng et al., 2017), bidirectional DAG LSTM (Bidir DAG LSTM) (Song et al., 2018b) and Graph State LSTM (GS GLSTM) (Song et al., 2018b). These methods extend LSTM to encode graphs constructed from input sentences with dependency edges, 3) Graph convolutional networks (GCN) with pruned trees, which have shown efficacy on the relation extraction task (Zhang et al., 2018)[8]. Addition-

---

[4]The number of entities is fixed in n-ary relation extraction task. It is 3 for the first dataset and 2 for the second.

[5]The dataset is available at https://github.com/freesunshine0316/nary-grn

[6]https://nlp.stanford.edu/projects/tacred/

[7]We use the 300-dimensional Glove word vectors trained on the Common Crawl corpus https://nlp.stanford.edu/projects/glove/

[8]The results are produced by the open implementation of Zhang et al. (2018).

Model	Binary-class				Multi-class	
	T		B		T	B
	Single	Cross	Single	Cross	Cross	Cross
Feature-Based (Quirk and Poon, 2017)	74.7	77.7	73.9	75.2	-	-
SPTree (Miwa and Bansal, 2016)	-	-	75.9	75.9	-	-
Graph LSTM-EMBED (Peng et al., 2017)	76.5	80.6	74.3	76.5	-	-
Graph LSTM-FULL (Peng et al., 2017)	77.9	80.7	75.6	76.7	-	-
+ multi-task	-	82.0	-	78.5	-	-
Bidir DAG LSTM (Song et al., 2018b)	75.6	77.3	76.9	76.4	51.7	50.7
GS GLSTM (Song et al., 2018b)	80.3	83.2	83.5	83.6	71.7	71.7
GCN (Full Tree) (Zhang et al., 2018)	84.3	84.8	84.2	83.6	77.5	74.3
GCN (K=0) (Zhang et al., 2018)	85.8	85.8	82.8	82.7	75.6	72.3
GCN (K=1) (Zhang et al., 2018)	85.4	85.7	83.5	83.4	78.1	73.6
GCN (K=2) (Zhang et al., 2018)	84.7	85.0	83.8	83.7	77.9	73.1
AGGCN (ours)	**87.1**	**87.0**	**85.2**	**85.6**	**79.7**	**77.4**

Table 1: Average test accuracies in five-fold validation for binary-class n-ary relation extraction and multi-class n-ary relation extraction. "T" and "B" denote ternary drug-gene-mutation interactions and binary drug-mutation interactions, respectively. Single means that we report the accuracy on instances within single sentences, while Cross means the accuracy on all instances. K in the GCN models means that the preprocessed pruned trees include tokens up to distance K away from the dependency path in the LCA subtree.

ally, we follow (Song et al., 2018b) to consider the tree-structured LSTM method (SPTree) (Miwa and Bansal, 2016) on drug-mutation binary relation extraction. Main results are shown in Table 1.

We first focus on the binary-class n-ary relation extraction task. For ternary relation extraction (first two columns in Table 1), our AGGCN model achieves accuracies of 87.1 and 87.0 on instances within single sentence (Single) and on all instances (Cross), respectively, which outperform all the baselines. More specifically, our AGGCN model surpasses the state-of-the-art Graph-structured LSTM model (GS GLSTM) by 6.8 and 3.8 points for the Single and Cross settings, respectively. Compared to GCN models , our model obtains 1.3 and 1.2 points higher than the best performing model with pruned tree (K=1). For binary relation extraction (third and fourth columns in Table 1), AGGCN consistently outperforms GS GLSTM and GCN as well.

These results suggest that, compared to previous full tree based methods, e.g., GS GLSTM, AGGCN is able to extract more information from the underlying graph structure to learn a more expressive representation through graph convolutions. AGGCN also performs better than GCNs, although its performance can be boosted via pruned trees. We believe this is because of the combination of densely connected layer and

attention guided layer. The dense connections could facilitate information propagation in large graphs, enabling AGGCN to efficiently learn from long-distance dependencies without pruning techniques. Meanwhile, the attention guided layer can further distill relevant information and filter out noises from the representation learned by the densely connected layer.

We next show the results on the multi-class classification task (last two columns in Table 1). We follow (Song et al., 2018b) to evaluate our model on all instances for both ternary and binary relations. This fine-grained classification task is much harder than coarse-grained classification task. As a result, the performance of all models degrades a lot. However, our AGGCN model still obtains 8.0 and 5.7 points higher than the GS GLSTM model for ternary and binary relations, respectively. We also notice that our AGGCN achieves a better test accuracy than all GCN models, which further demonstrates its ability to learn better representations from full trees.

3.4 Results on Sentence-level Relation Extraction

We now report the results on the TACRED dataset for the sentence-level relation extraction task in Table 2. We compare our model against two kinds of models: 1) dependency-based models, 2)

Model	P	R	F1
LR (Zhang et al., 2017)	**73.5**	49.9	59.4
SDP-LSTM (Xu et al., 2015c)*	66.3	52.7	58.7
Tree-LSTM (Tai et al., 2015)**	66.0	59.2	62.4
PA-LSTM (Zhang et al., 2017)	65.7	64.5	65.1
GCN (Zhang et al., 2018)	69.8	59.0	64.0
C-GCN (Zhang et al., 2018)	69.9	63.3	66.4
AGGCN (ours)	69.9	60.9	65.1
C-AGGCN (ours)	71.8	**66.4**	**69.0**

Table 2: Results on the TACRED dataset. Model with * indicates that the results are reported in Zhang et al. (2017), while model with ** indicates the results are reported in Zhang et al. (2018).

Model	F1
SVM (Rink and Harabagiu, 2010)	82.2
SDP-LSTM (Xu et al., 2015c)	83.7
SPTree (Miwa and Bansal, 2016)	84.4
PA-LSTM (Zhang et al., 2017)	82.7
C-GCN (Zhang et al., 2018)	84.8
C-AGGCN (ours)	**85.7**

Table 3: Results on the SemEval dataset.

sequence-based models. Dependency-based models include the logistic regression classifier (LR) (Zhang et al., 2017), Shortest Path LSTM (SDP-LSTM) (Xu et al., 2015c), Tree-structured neural model (Tree-LSTM) (Tai et al., 2015), GCN and Contextualized GCN (C-GCN) (Zhang et al., 2018). Both GCN and C-GCN models use the pruned trees. For sequence-based model, we consider the state-of-the-art Position Aware LSTM (PA-LSTM) (Zhang et al., 2017).

As shown in Table 2, the logistic regression classifier (LR) obtains the highest precision score. We hypothesize that the reason behind this is due to the data imbalance issue. This feature-based method tends to predict a highly frequent label as the relation (e.g., "per:title"). Therefore, it has a high precision while having a relatively low recall. On the other hand, the neural models are able to better balance the precision and recall scores.

Since GCN and C-GCN already show their superiority over other dependency-based models and PA-LSTM, we mainly compare our AGGCN model with them. We can observe that AGGCN outperforms GCN by 1.1 F1 points. We speculate

Model	F1
C-AGGCN	69.0
– Attention-guided layer (AG)	67.1
– Dense connected layer (DC)	67.3
– AG, DC	66.7
– Feed-Forward layer (FF)	67.8

Table 4: An ablation study for C-AGGCN model.

Model	F1
C-AGGCN (Full tree)	69.0
C-AGGCN (K=2)	67.5
C-AGGCN (K=1)	67.9
C-AGGCN (K=0)	67.0

Table 5: Results of C-AGGCN with pruned trees.

that the limited improvement is due to the lack of contextual information about word order or disambiguation. Similar to C-GCN (Zhang et al., 2018), we extend our AGGCN model with a bi-directional LSTM network to capture the contextual representations which are subsequently fed into AGGCN layers. We term the modified model as C-AGGCN. Our C-AGGCN model achieves an F1 score of 69.0, which outperforms the state-of-art C-GCN model by 2.6 points. We also notice that AGGCN and C-AGGCN achieve better precision and recall scores than GCN and C-GCN, respectively. The performance gap between GCNs with pruned trees and AGGCNs with full trees empirically show that the AGGCN model is better at distinguishing relevant from irrelevant information for learning a better graph representation.

We also evaluate our model on the SemEval dataset under the same settings as (Zhang et al., 2018). Results are shown in Table 3. This dataset is much smaller than TACRED (only 1/10 of TACRED in terms of the number of instances). Our C-AGGCN model (85.7) consistently outperforms the C-GCN model (84.8), showing the good generalizability.

3.5 Analysis and Discussion

Ablation Study. We examine the contributions of two main components, namely, densely connected layers and attention guided layers, using the best-performing C-AGGCN model on the TACRED dataset. Table 4 shows the results. We can observe that adding either attention guided layers
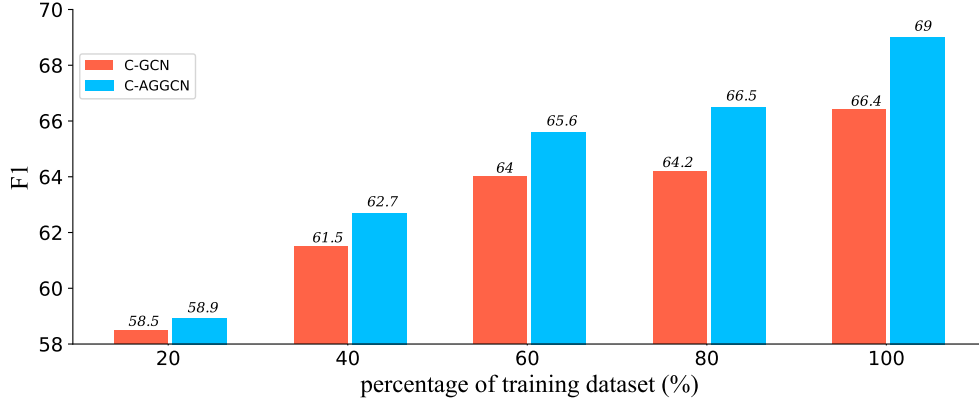
Figure 3: Comparison of C-AGGCN and C-GCN against different training data sizes. The results of C-GCN are reproduced from (Zhang et al., 2018).
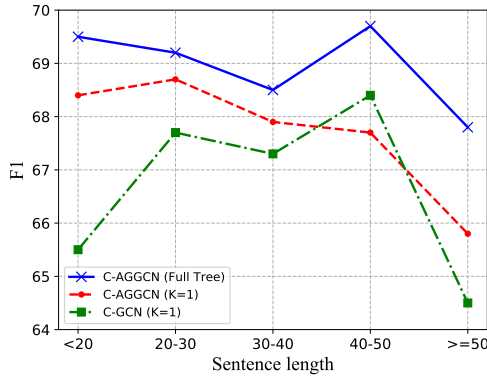

Figure 4: Comparison of C-AGGCN and C-GCN against different sentence lengths. The results of C-GCN are reproduced from (Zhang et al., 2018).

or densely connected layers improves the performance of the model. This suggests that both layers can assist GCNs to learn better information aggregations, producing better representations for graphs, where the attention-guided layer seems to be playing a more significant role. We also notice that the feed-forward layer is effective in our model. Without the feed-forward layer, the result drops to an F1 score of 67.8.

Performance with Pruned Trees. Table 5 shows the performance of the C-AGGCN model with pruned trees, where K means that the pruned trees include tokens that are up to distance K away from the dependency path in the LCA subtree. We can observe that all the C-AGGCN models with varied values of K are able to outperform the state-of-the-art C-GCN model (Zhang et al., 2018) (reported in Table 2). Specifically, with the same setting as K=1, C-AGGCN surpasses C-GCN by

1.5 points of F1 score. This demonstrates that, with the combination of densely connected layer and attention guided layer, C-AGGCN can learn better representations of graphs than C-GCN for downstream tasks. In addition, we notice that the performance of C-AGGCN with full trees outperforms all C-AGGCNs with pruned trees. These results further show the superiority of "soft pruning" strategy over hard pruning strategy in utilizing full tree information.

Performance against Sentence Length. Figure 4 shows the F1 scores of three models under different sentence lengths. We partition the sentence length into five classes (< 20, $[20, 30)$, $[30, 40)$, $[40, 50)$, ≥ 50). In general, C-AGGCN with full trees outperforms C-AGGCN with pruned trees and C-GCN against various sentence lengths. We also notice that C-AGGCN with pruned trees performs better than C-GCN in most cases. Moreover, the improvement achieved by C-AGGCN with pruned trees decays when the sentence length increases. Such a performance degradation can be avoided by using full trees, which provide more information of the underlying graph structures. Intuitively, with the increase of the sentence length, the dependency graph becomes larger as more nodes are included. This suggests that C-AGGCN can benefit more from larger graphs (full tree).

Performance against Training Data Size. Figure 3 shows the performance of C-AGGCN and C-GCN against different settings for training with different amount of training data. We consider five training settings (20%, 40%, 60%, 80%, 100% of the training data). C-AGGCN consistently outper-

forms C-GCN under the same amount of training data. When the size of training data increases, we can observe that the performance gap becomes more obvious. Specifically, using 80% of the training data, the C-AGGCN model is able to achieve a F1 score of 66.5, higher than C-GCN trained on the complete training set. These results demonstrate that our model is more effective in terms of using training resources.

4 Related Work

Our work builds on a rich line of recent efforts on relation extraction models and graph convolutional networks.

Relation Extraction. Early research efforts are based on statistical methods. Tree-based kernels (Zelenko et al., 2002) and dependency path-based kernels (Bunescu and Mooney, 2005) are explored to extract the relation. McDonald et al. (2005) construct maximal cliques of entities to predict relations. Mintz et al. (2009) include syntactic features to a statistical classifier. Recently, sequence-based models leverages different neural networks to extract relations, including convolutional neural networks (Zeng et al., 2014; Nguyen and Grishman, 2015; Wang et al., 2016), recurrent neural networks (Zhou et al., 2016; Zhang et al., 2017) the combination of both (Vu et al., 2016) and transformer (Verga et al., 2018).

Dependency-based approaches also try to incorporate structural information into the neural models. Peng et al. (2017) first split the dependency graph into two DAGs, then extend the tree LSTM model (Tai et al., 2015) over these two graphs for n-ary relation extraction. Closest to our work, Song et al. (2018b) use graph recurrent networks (Song et al., 2018a) to directly encode the whole dependency graph without breaking it. The contrast between our model and theirs is reminiscent of the contrast between CNN and RNN. Various pruning strategies have also been proposed to distill the dependency information in order to further improve the performance. Xu et al. (2015b,c) adapt neural models to encode the shortest dependency path. Miwa and Bansal (2016) apply LSTM model over the LCA subtree of two entities. Liu et al. (2015) combine the shortest dependency path and the dependency subtree. Zhang et al. (2018) adopt a path-centric pruning strategy. Unlike these strategies that remove edges in preprocessing, our model learns to assign each edge a different weight in an end-to-end fashion.

Graph Convolutional Networks. Early efforts that attempt to extend neural networks to deal with arbitrary structured graphs are introduced by Gori et al. (2005); Bruna (2014). Subsequent efforts improve its computational efficiency with local spectral convolution techniques (Henaff et al., 2015; Defferrard et al., 2016). Our approach is closely related to the GCNs (Kipf and Welling, 2017), which restrict the filters to operate on a first-order neighborhood around each node.

More recently, Velickovic et al. (2018) proposed graph attention networks (GATs) to summarize neighborhood states by using masked self-attentional layers (Vaswani et al., 2017). Compared to our work, their motivations and network structures are different. In particular, each node only attends to its neighbors in GATs whereas AG-GCNs measure the relatedness among all nodes. The network topology in GATs remains the same, while fully connected graphs will be built in AG-GCNs to capture long-range semantic interactions.

5 Conclusion

We introduce the novel Attention Guided Graph Convolutional Networks (AGGCNs). Experimental results show that AGGCNs achieve state-of-the-art results on various relation extraction tasks. Unlike previous approaches, AGGCNs operate directly on the full tree and learn to distill the useful information from it in an end-to-end fashion. There are multiple venues for future work. One natural question we would like to ask is how to make use of the proposed framework to perform improved graph representation learning for graph related tasks (Bastings et al., 2017).

Acknowledgements

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proc. of EMNLP*.

Joan Bruna. 2014. Spectral networks and deep locally connected networks on graphs. In *Proc. of ICLR*.

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

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proc. of EMNLP*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. of NeurIPS*.

Michele Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proc. of IJCNN*.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association of Computational Linguistics*.

Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *SemEval@ACL*.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *Proc. of CVPR*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proc. of ICLR*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proc. of EMNLP*.

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

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proc. of EMNLP*.

Ryan T. McDonald, Fernando Pereira, Seth Kulick, R. Scott Winters, Yang Jin, and Peter S. White. 2005. Simple algorithms for complex relation extraction with applications to biomedical ie. In *Proc. of ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proc. of ACL*.

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

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proc. of VS@NAACL-HLT*.

Adam Paszke, Sam Gross, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *Proc. of workshop on NeurIPS*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115.

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

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proc. of EACL*.

Bryan Rink and Sanda M. Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *SemEval@ACL*.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Timothy P. Lillicrap. 2017. A simple neural network module for relational reasoning. In *Proc. of NeurIPS*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018a. A graph-to-sequence model for amr-to-text generation. In *Proc. of ACL*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018b. N-ary relation extraction using graph state lstm. In *Proc. of EMNLP*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. of ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proc. of ICLR*.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proc. of NAACL-HLT*.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proc. of NAACL-HLT*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proc. of ACL*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015a. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. of ICML*.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *Proc. of ICML*.

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

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015c. Classifying relations via long short term memory networks along shortest dependency paths. In *Proc. of EMNLP*.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proc. of ACL*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proc. of EMNLP*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jian Zhao. 2014. Relation classification via convolutional deep neural network. In *Proc. of COLING*.

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

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proc. of EMNLP*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proc. of ACL*.