Graph Convolution over Pruned Dependency Trees Improves Relation Extraction

Yuhao Zhang,* Peng Qi,* Christopher D. Manning
Stanford University
Stanford, CA 94305
{yuhao, pengqi, manning}@cs.stanford.edu

Abstract

Dependency trees help relation extraction models capture long-range relations between words. However, existing dependency-based models either neglect crucial information (e.g., negation) by pruning the dependency trees too aggressively, or are computationally inefficient because they involve propagating information sequentially over the entire tree. We propose a novel neural architecture based on graph convolutional networks that pools information over arbitrary dependency structures efficiently in parallel. To incorporate relevant information while maximally removing irrelevant content, we apply a novel pruning strategy to the input trees by keeping words immediately around the shortest path between the two entities among which a relation might hold. The resulting model achieves state-of-the-art performance on the large-scale TACRED dataset, outperforming existing sequence and dependency-based neural models. We also find that this model has complementary strengths with sequence models, and combining them further improves the state of the art.

1 Introduction

Relation extraction involves discerning whether a relation exists between two entities in a sentence (often termed *subject* and *object*, respectively). Successful relation extraction is the cornerstone of applications requiring relational understanding of unstructured text on a large scale, such as question answering (Yu et al., 2017), knowledge base population (Zhang et al., 2017), and biomedical knowledge discovery (Quirk and Poon, 2017).

I had an e-mail exchange with Benjamin Cane of Popular Mechanics which showed that **he** was not a close relative of ***Mike Cane***.

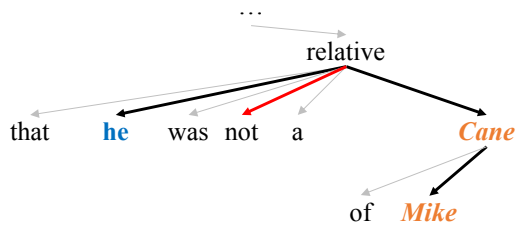

Figure 1: An example sentence modified from the TAC KBP challenge corpus. A subtree of the original UD dependency tree between the subject ("he") and object ("Mike Cane") is also shown, where the shortest dependency path between the two entities is highlighted in bold. Note that negation ("not") is off the shortest dependency path.

Models making use of dependency parses of the input sentences, or *dependency-based models*, have proven very effective in relation extraction, because they capture long-range relations that are obscure from the surface form alone (e.g., when long clauses or complex scoping are present). To represent dependency information, traditional feature-based models featurize dependency trees as overlapping paths along the trees to train classifiers over them (Kambhatla, 2004). However, these models face the challenge of sparse feature spaces and are brittle to lexical variations. More recent neural models address this problem with distributed representations built from their computation graphs formed along parse trees. One common approach to leverage dependency information is to perform bottom-up or top-down computation along the parse tree or the subtree below the lowest common ancestor (LCA) of the subject and object

*Equal contribution. The order of authorship was decided by a tossed coin.

entities (Miwa and Bansal, 2016). Another popular approach, inspired by Bunescu and Mooney (2005), is to reduce the parse tree to the shortest path between the entities (Xu et al., 2015a,b).

However, these models suffer from several drawbacks. Neural models operating directly on parse trees are usually difficult to parallelize and thus computationally inefficient, because they propagate information sequentially over tree structures. Models based on the shortest *dependency path* between the subject and object are computationally more efficient by considering fewer words, but this simplifying assumption has major limitations as well. Figure 1 shows a real-world example where crucial information (i.e., negation) would be excluded when the model is restricted to only considering the dependency path.

Inspired by the recent work on graph convolutional networks (Kipf and Welling, 2017; Marcheggiani and Titov, 2017), we propose a novel neural architecture that is tailored for relation extraction. Our model encodes the dependency structure over the input sentence with efficient graph convolution operations, then extracts entity-centric representations to make robust relation predictions. We also apply a novel *path-centric pruning* technique to remove irrelevant information from the tree while maximally keeping relevant content, which further improves the performance of several dependency-based models including ours.

We test our model's performance on the popular SemEval 2010 Task 8 dataset and the more recent, large-scale TACRED dataset. On both datasets, our model outperforms existing dependency-based neural models by a substantial margin when combined with the new pruning technique. On TACRED, our model further achieves the state-of-the-art performance, surpassing a competitive neural sequence model baseline. The proposed model also exhibits complementary strengths to sequence models on TACRED, and an ensemble of these two model types through simple prediction interpolation further improves the state of the art.

To recap, our main contributions in this paper are: (i) we propose a novel neural architecture for relation extraction based on graph convolutional networks, which allows it to efficiently pool dependency information over arbitrary tree structure; (ii) we present a new *path-centric pruning* technique to help dependency-based models maximally remove irrelevant information without damaging crucial content to improve their robustness.

2 Related Work

At the core of fully-supervised and distantly-supervised relation extraction approaches are statistical classifiers, many of which find syntactic information beneficial. For example, Mintz et al. (2009) explored adding syntactic features to a statistical classifier and found them to be especially useful when sentences are long. Various kernel-based approaches also leverage dependency information to measure similarity between training and test examples to predict the relation, finding that tree-base kernels (Zelenko et al., 2003) and dependency path-based kernels (Bunescu and Mooney, 2005) are effective for this task.

Recent studies have found neural models effective in relation extraction. Zeng et al. (2014) first applied a one-dimensional convolutional neural network (CNN) with hand-tuned features to encode relations. Vu et al. (2016) showed that combining a CNN with a recurrent neural network (RNN) through a voting scheme can further improve performance. Zhou et al. (2016) and Wang et al. (2016) proposed to use attention mechanisms over RNN and CNN architectures for this task.

Apart from neural models over word sequences, incorporating dependency trees into neural models has also been shown to improve relation extraction performance by capturing long-term relations. Xu et al. (2015b) generalized the idea of dependency path kernels by applying a long short-term memory (LSTM) network over the shortest dependency path between entities. Liu et al. (2016) showed that augmenting dependency paths with additional words by following rules can further improve performance. Miwa and Bansal (2016) proposed to apply a Tree-LSTM (Tai et al., 2015), a generalized form of LSTM over dependency trees, in a joint entity and relation extraction setting. They found it to be most effective when applied to the subtree rooted at the LCA of the two entities.

More recently, Adel et al. (2016) and Zhang et al. (2017) have shown that relatively simple neural models (CNN and modified LSTM, respectively) can achieve comparable or superior performance to dependency-based models when trained on larger datasets. In this paper, we study dependency-based models in depth and present that with a properly designed architecture, they

can outperform and have complementary advantages to sequence models, even in a large-scale setting.

3 Models

In this section, we first describe graph convolutional network (GCN) over dependency tree structures, then we introduce an architecture that uses GCN at its core for relation extraction.

3.1 Graph Convolutional Networks over Dependency Trees

The graph convolutional network (Kipf and Welling, 2017, GCN) is an adaptation of the convolutional neural network (LeCun et al., 1998) for encoding graphs. Given a graph with n nodes, we can represent the graph structure with an $n \times n$ adjacency matrix $\mathbf{A}$ where $A_{ij} = 1$ if there is an edge going from node i to node j. In an L-layer GCN, if we denote by $h_i^{(l-1)}$ the input vector and $h_i^{(l)}$ the output vector of node i at the l-th layer, a graph convolutional operation can be written as

$$h_i^{(l)} = \sigma\big(\sum_{j=1}^{n} A_{ij} W^{(l)} h_j^{(l-1)} + b^{(l)}\big), \quad (1)$$

where $W^{(l)}$ is a linear transformation, $b^{(l)}$ a bias term, and σ a nonlinearity function (e.g., ReLU). Intuitively, during each graph convolution, each node gathers and summarizes information from its neighboring nodes in the graph.

This operation can be naturally adapted to encode dependency information between tokens within a sentence. Given a sentence with n tokens and its dependency parse tree where each node represents a token, we convert the tree into its corresponding adjacency matrix $\mathbf{A}$ where $A_{ij} = 1$ if there is a dependency edge between tokens i and j. Marcheggiani and Titov (2017) and Bastings et al. (2017) have found it beneficial to consider the dependency parse as a directed graph and parameterize the top-down and bottom-up directions separately in semantic role labeling and machine translation, respectively. In this work, we find it sufficient to treat the dependency tree as an undirected graph, i.e. $\forall i, j, A_{ij} = A_{ji}$.

Naively applying the graph convolutional operation in Equation (1) to dependency trees has two drawbacks. First, since a token never connects to itself in a dependency tree, the information in $h_i^{(l-1)}$ is lost in this operation and never carried over to $h_i^{(l)}$. Second, since the degree of a token varies a lot in the dependency tree, taking the sum of representations over all neighboring tokens without normalization will result in token representations of very different magnitudes, leading to undesirable numerical behaviors.

Inspired by (Kipf and Welling, 2017), we resolve these issues by adding a self-loop to each node and normalizing activations as follows:

$$\tilde{h}_i^{(l)} = W^{(l)} h_i^{(l-1)} + \sum_{j=1}^{n} A_{ij} W^{(l)} h_j^{(l-1)}, \quad (2)$$

$$h_i^{(l)} = \sigma\big(\tilde{h}_i^{(l)} / (d_i + 1) + b^{(l)}\big). \quad (3)$$

where $d_i = \sum_{j=1}^{n} A_{ij}$ is the degree of token i in the tree. Here, we simply average the transformed representations of each token in token i's neighborhood before applying the nonlinearity. In our experiments, we find this simple normalization to work better than the two-sided normalization in (Kipf and Welling, 2017) as well as not using any normalization in (Marcheggiani and Titov, 2017).

Stacking this operation over L layers gives us a deep GCN network, where we set $h_1^{(0)}, \ldots, h_n^{(0)}$ to be input word vectors, and use $h_1^{(L)}, \ldots, h_n^{(L)}$ as output word representations. Note that if we stack the input and output representations into matrices, graph convolution can be simply written as

$$\mathbf{h}^{(l)} = \sigma\big(W^{(l)} \mathbf{h}^{(l-1)} (\mathbf{A} + \mathbf{I}) \mathbf{D}^{-1} + b^{(l)} \otimes 1_n\big), \quad (4)$$

where $\mathbf{I}$ is the $n \times n$ identity matrix, $\mathbf{D}$ is the diagonal matrix where $D_{ii} = d_i + 1, \forall i$, and 1_n represents the n-dimensional vector with all ones.

Note that all operations in Equation (4) can be efficiently implemented with matrix multiplications, making it ideal for batching computation over examples and running on GPUs. Moreover, the propagation of information between tokens occurs in parallel, and the running time does not depend on the depth of the original dependency tree.

3.2 Encoding Relations with GCN

We now formally define the task of relation extraction. Let $\mathcal{X} = [x_1, ..., x_n]$ denote a sentence, where x_i is the i-th token. A subject entity and an object entity are identified and correspond to two spans in the sentence: $\mathcal{X}_s = [x_{s_1}, \ldots, x_{s_2}]$ and $\mathcal{X}_o = [x_{o_1}, \ldots, x_{o_2}]$. Given $\mathcal{X}$, $\mathcal{X}_s$, and $\mathcal{X}_o$, the goal of relation extraction is to predict a relation $r \in \mathcal{R}$ (a predefined relation set) that holds between the entities or "no relation" otherwise.

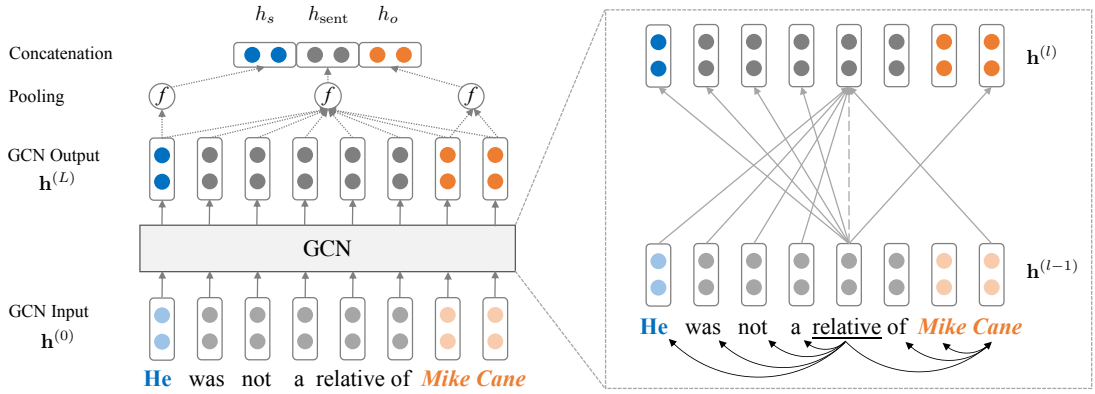

Figure 2: Relation extraction with a graph convolutional network. In the graph convolution visualization on the right, we only show the operation around the word "relative" for clarity. A full unlabeled dependency parse of the sentence is also provided for reference.

After applying an L-layer GCN over word vectors, we obtain hidden representations of each token that are directly influenced by its neighbors no more than L edges apart in the dependency tree. To make use of these word representations for relation extraction, we first obtain a sentence representation as follows (see also Figure 2 left):

$$h_{\text{sent}} = f\big(\mathbf{h}^{(L)}\big) = f\big(\text{GCN}(\mathbf{h}^{(0)})\big), \quad (5)$$

where $f : \mathbb{R}^{d\times n} \rightarrow \mathbb{R}^d$ is a pooling function that maps from n output vectors to the sentence vector.

We also observe that information close to entity tokens in the dependency tree is often central to relation classification. Therefore, we also obtain a subject representation h_s from $\mathbf{h}^{(L)}$ as follows

$$h_s = f\big(\mathbf{h}^{(L)}_{s_1:s_2}\big), \quad (6)$$

as well as an object representation h_o similarly.

Inspired by recent work on relational learning between entities (Santoro et al., 2017; Lee et al., 2017), we obtain the final representation used for classification by concatenating the sentence and the entity representations, and feeding them through a feed-forward neural network (FFNN):

$$h_{\text{final}} = \text{FFNN}\big([h_{\text{sent}}; h_s; h_o]\big). \quad (7)$$

This h_{final} representation is then fed into a linear layer followed by a softmax operation to obtain a probability distribution over relations.

3.3 Contextualized GCN

The network architecture introduced so far learns effective representations for relation extraction, but it also leaves a few issues inadequately addressed. First, the input word vectors do not contain contextual information about word order or disambiguation. Second, the GCN highly depends on a correct parse tree to extract crucial information from the sentence (especially when pruning is performed), while existing parsing algorithms produce imperfect trees in many cases.

To resolve these issues, we further propose a Contextualized GCN (C-GCN) model, where the input word vectors are first fed into a bi-directional LSTM network to generate contextualized representations, which are then used as $\mathbf{h}^{(0)}$ in the original model. We show empirically in Section 5 that this augmentation further improves the performance over the original model.

We note that our proposed model is conceptually similar to graph kernel-based models for relation extraction (Zelenko et al., 2003), in that it aims to utilize local dependency tree patterns to inform relation classification. Our model also incorporates crucial off-path information, which greatly improves its robustness compared to shortest dependency path-based approaches. Compared to tree-structured models (e.g., Tree-LSTM), our model does not require recursive propagation of information through tree branches, and is able to capture more global information through the use of pooling functions. More importantly, in practice we find that our GCN model can achieve a 10–100× speed up compared to Tree-LSTM models, which are difficult to parallelize.

4 Incorporating Off-path Information with Path-centric Pruning

Dependency trees provide rich structures that one can exploit in relation extraction, but most of the information pertinent to relations is usually con-

tained within the subtree rooted at the lowest common ancestor (LCA) of the two entities. Previous studies (Xu et al., 2015b; Miwa and Bansal, 2016) have shown that removing tokens outside this scope helps relation extraction by eliminating irrelevant information from the sentence. It is therefore desirable to combine our GCN models with tree pruning strategies to further improve performance. However, pruning too aggressively (e.g., keeping only the dependency path) could lead to loss of crucial information and conversely hurt robustness. For instance, the negation in Figure 1 is neglected when a model is restricted to only looking at the dependency path between the entities. Similarly, in the sentence "*She was diagnosed with cancer last year, and succumbed this June*", the dependency path *She*←*diagnosed*→*cancer* is not sufficient to establish that *cancer* is the cause of death for the subject unless the conjunction dependency to *succumbed* is also present.

Motivated by these observations, we propose *path-centric pruning*, a novel technique to incorporate information off the dependency path. This is achieved by including tokens that are up to distance K away from the dependency path in the LCA subtree. $K = 0$, for instance, corresponds to pruning the tree down to the dependency path, and $K = \infty$ retains the entire LCA subtree. We combine this pruning strategy with our GCN model, by directly feeding the pruned trees into the graph convolutional layers.[1] We hypothesize that pruning with $K = 1$ achieves the best balance between including relevant information (e.g., negation and conjunction) and keeping irrelevant content out of the resulting pruned tree as much as possible.

5 Experiments

5.1 Baseline Models

We compare our models with several competitive dependency-based and neural sequence models.

Dependency-based models. We compare with three types of dependency-based models: (1) Logistic regression (LR) classifier which combines dependency-based features with other lexical features. (2) Shortest Dependency Path LSTM (Xu et al., 2015b, SDP-LSTM), which applies a neural sequence model on the shortest path between the subject and object entities in the dependency tree. (3) Tree-LSTM (Tai et al., 2015), which is a recursive model that generalizes the LSTM to arbitrary tree structures. We investigate the child-sum variant of Tree-LSTM, and apply it to the dependency tree (or part of it). In practice, we find that modifying this model by concatenating dependency label embeddings to the input of forget gates improves its performance on relation extraction, and therefore use this variant in our experiments.[2] Zhang et al. (2017) compared (1) and (2) with sequence models, and we report their results; for (3) we report results with our own implementation.

Neural sequence model. Zhang et al. (2017) presented a competitive sequence model that employs a position-aware attention mechanism over LSTM outputs (PA-LSTM), and showed that it outperforms several CNN and dependency-based models by a substantial margin. We compare with this strong baseline, and use our open implementation in further analysis.[3]

5.2 Experimental Setup

We conduct experiments on two relation extraction datasets: (1) **TACRED**: TACRED is a large-scale relation extraction dataset introduced in (Zhang et al., 2017), containing over 106k mention pairs drawn from the yearly TAC KBP challenge. It represents 41 relation types and a special *no_relation* class when the mention pair does not have a relation between them within these categories. Mentions in TACRED are typed, with subjects categorized into person and organization, and objects into 16 fine-grained types (e.g., date and location). We report micro-averaged F_1 scores on this dataset as is conventional. (2) **SemEval 2010 Task 8**: The SemEval dataset is widely used in previous work, but is significantly smaller with 8,000 examples for training and 2,717 for testing. It contains 19 relation classes over untyped mention pairs: 9 directed relations and a special *Other* class. On SemEval, we follow convention and report the official macro-averaged F_1 scores.

For fair comparisons on the TACRED dataset, we follow the evaluation protocol used in (Zhang et al., 2017) by selecting the model with the median dev F_1 from 5 independent runs and reporting its test F_1. We also use the same "entity mask"

[1] For our C-GCN model, the LSTM layer still operates on the full sentence regardless of the pruning.

[2] We hypothesize that this variant allows the model to disregard non-crucial subtrees such as determiners ("the", "a").

[3] https://github.com/yuhaozhang/tacred-relation

System	P	R	F_1
LR* (Zhang+2017)	**73.5**	49.9	59.4
SDP-LSTM* (Xu+2015b)	66.3	52.7	58.7
Tree-LSTM† (Tai+2015)	66.0	59.2	62.4
PA-LSTM* (Zhang+2017)	65.7	64.5	65.1
GCN	68.5	59.3	63.6
GCN + PA-LSTM	70.4	63.7	66.9
C-GCN	69.0	63.0	65.8
C-GCN + PA-LSTM	70.3	**64.9**	**67.5**

Table 1: Results on the TACRED test set. Underscore indicates highest number among single models; bold indicates highest among all. $*$ indicates results reported in (Zhang et al., 2017); † indicates results produced with our implementation.

System	F_1	F_1 (mask)
SVM* (Rink+2010)	82.2	–
SDP-LSTM* (Xu+2015b)	83.7	–
SPTree* (Miwa+2016)	84.4	–
PA-LSTM† (Zhang+2017)	82.7	75.3
Our Model (C-GCN)	**84.8**	**76.5**

Table 2: Results on the SemEval test set. $*$ indicates results reported in the original papers; † indicates results produced by using our open implementation. The middle column shows results from *with-entity* evaluation; the right column shows results from *mask-entity* evaluation.

strategy where we replace each subject (and object similarly) entity with a special *SUBJ-<NER>* token. For all models, we also adopt the "multi-channel" strategy by concatenating the input word embeddings with POS and NER embeddings.

Traditionally, evaluation on SemEval is conducted without entity tokens masked. However, as we will discuss in Section 6.4, this method encourages models to overfit to entity tokens and fails to test their actual ability to generalize. We therefore report results with two evaluation protocols: (1) *with-entity*, where entity tokens are kept for comparison with previous work; and (2) *mask-entity*, where entity tokens are masked to test the generalization of our model in a more realistic setting.

Due to space limit, we report hyperparameters and training details in the supplementary material.

5.3 Results on the TACRED Dataset

We present our main results on the TACRED test set in Table 1. We observe that our GCN model outperforms all dependency-based models substantially. By using contextualized word representations, the C-GCN model further outperforms the strong PA-LSTM model by 0.7 F_1, and achieves a new state of the art. In addition, we find our models improve upon other dependency-based models in both precision and recall. Comparing the C-GCN model with the GCN model, we find that the gain mainly comes from improved recall.

As we will show in Section 6.2, we find that our GCN models have complementary strengths when compared to the PA-LSTM. To leverage this result, we experiment with a simple interpolation strategy to ensemble these models. Given the output probabilities $P_G(r|x)$ from a GCN model and $P_S(r|x)$ from the sequence model for any relation r, we calculate the interpolated probability as

$$P(r|x) = \alpha \cdot P_G(r|x) + (1 - \alpha) \cdot P_S(r|x)$$

where $\alpha \in [0, 1]$ is chosen on the dev set. We find $\alpha = 0.5$ to be most effective. This simple interpolation between a GCN and a PA-LSTM achieves an F_1 score of 66.9, very close to the F_1 score of 67.2 as reported in (Zhang et al., 2017) from ensembling 5 PA-LSTM models. An interpolation between a C-GCN and a PA-LSTM further achieves a new state-of-the-art result of 67.5.

5.4 Results on the SemEval Dataset

To study the generalizability of our proposed model, we also trained and evaluated our best C-GCN model on the SemEval dataset (Table 2). We find that under the conventional *with-entity* evaluation, our C-GCN model outperforms all existing dependency-based neural models on this separate dataset. Notably, by properly incorporating off-path information, our model substantially outperforms the previous shortest dependency path-based model (SDP-LSTM). Under the *mask-entity* evaluation, our C-GCN model still outperforms PA-LSTM by a substantial margin, suggesting its generalizability even when entities are not seen.

5.5 Effect of Path-centric Pruning

To show the effectiveness of path-centric pruning, we compare the two GCN models and the Tree-LSTM when the pruning distance K is varied. We experimented with $K \in \{0, 1, 2, \infty\}$ on the TACRED dev set, and also include results when the full tree is used. As shown in Figure 3,

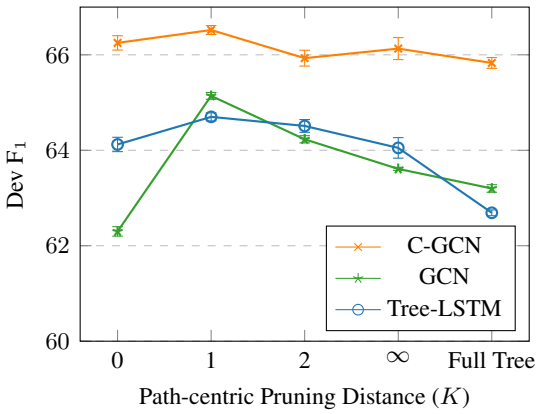

Figure 3: Performance of dependency-based models under different pruning strategies. For each model we show the F_1 score on the TACRED dev set averaged over 5 runs, and error bars indicate standard deviation of the mean estimate.

Model	Dev F_1
Best C-GCN	66.4
– Feedforward (FF) Layers	66.1
– LSTM Layer	65.1
– FF and LSTM	64.7
– Dependency	63.4
– FF, LSTM and Dependency	40.8

Table 3: An ablation study of the best C-GCN model. Scores are median of 5 models.

the performance of all three models peaks when $K = 1$, outperforming their respective dependency path-based counterpart ($K = 0$). This confirms our hypothesis in Section 4 that incorporating off-path information is crucial to relation extraction. Miwa and Bansal (2016) reported that a Tree-LSTM achieves similar performance when the dependency path and the LCA subtree are used respectively. Our experiments confirm this, and further show that the result can be improved by path-centric pruning with $K = 1$.

We find that all three models are less effective when the entire dependency tree is present, indicating that including extra information hurts performance. Finally, we note that contextualizing the GCN makes it less sensitive to changes in the tree structure provided, presumably because the model can use word sequence information in the LSTM layer to recover any off-path information that it needs for correct relation extraction.

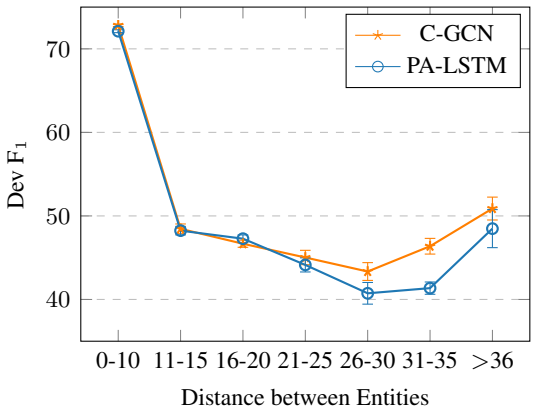

Figure 4: Dev set performance with regard to distance between the entities in the sentence for PA-LSTM and C-GCN. Error bars indicate standard deviation of the mean estimate over 5 runs.

6 Analysis & Discussions

6.1 Ablation Study

To study the contribution of each component in the C-GCN model, we ran an ablation study on the TACRED dev set (Table 3). We find that the feedforward layer and the LSTM layer contributes 0.3 and 1.3 F_1 respectively. The result drops by 1.7 F_1 when we remove them together. To study how much contribution comes from utilizing the dependency structures, we set the adjacency matrix to a zero matrix (– Dependency), and the F_1 drops by 3.0. This is equivalent to having an LSTM model with additional non-linear feedforward layers, and the result is close to what Zhang et al. (2017) reported from an LSTM model. Finally, removing these three components together removes any contextualization or information about the dependency parse, which greatly hampers the model's ability to discern relation in the sentence.

6.2 Complementary Strengths of GCNs and PA-LSTMs

To understand what the GCN models are capturing and how they differ from a sequence model such as the PA-LSTM, we compared their performance over examples in the TACRED dev set. Specifically, for each model, we trained it for 5 independent runs with different seeds, and for each example we evaluated the model's accuracy over these 5 runs. For instance, if a model correctly classifies an example for 3 out of 5 times, it achieves an accuracy of 60% on this example. We observe that on 906 (4.0%) dev examples, our C-GCN model achieves an accuracy at least 60% higher than that

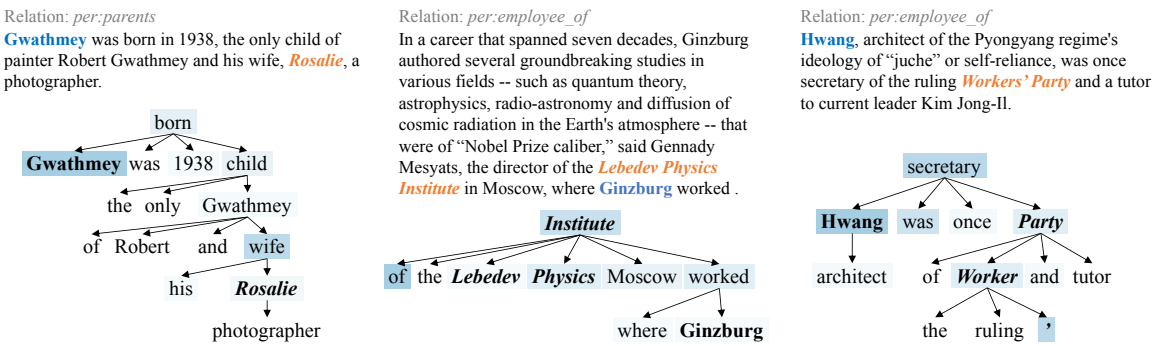

Figure 5: Examples that the C-GCN predicted correctly and the PA-LSTM misclassified. Words contributing to h_{sent} after pooling are highlighted (with punctuations omitted), shaded by the number of dimensions each word contributed. The first example shows that the model uses off-path information to improve confidence about its prediction and override distractor words ("wife" is usually an indicator of *per:spouse*), the second illustrates that pruning helps drastically narrow the model's focus, and the third shows the model captures a long-range relation between words even in the presence of parse errors.

of the PA-LSTM, while on 838 (3.7%) examples the PA-LSTM achieves 60% higher. This complementary performance explains the gain we see in Table 1 when the two models are ensembled.

We further show that this difference is likely due to each model's competitive advantage (Figure 4): dependency-based models are good at handling sentences with entities farther apart, while sequence models can better leverage local word patterns regardless of the quality of the parse. We also show examples where C-GCN outperforms PA-LSTM in Figure 5. We include further analysis in the supplementary material.

6.3 Understanding Model Behavior

To gain more insight into the C-GCN model's behavior, we visualized the partial dependency tree it is processing and how much each token's final representation contributed to h_{sent} (Figure 5).

We find that the model often focuses on the dependency path, but sometimes incorporates off-path information to help reduce the effect of distractor words on the path. The model also learns to ignore determiners (e.g., "the") and passive voice auxiliary verbs (e.g., "was" in "was born") as they rarely affect relation prediction. Finally, we find that subject and object tokens contribute substantially to the GCN's final representation, confirming our observation that local regions around these entities are central to relation classification.

6.4 Entity Bias in the SemEval Dataset

In our study, we observed a high correlation between the entities in a sentence and its relation label in the SemEval dataset. We experimented with PA-LSTM models to analyze this phenomenon.[4] We started by simplifying every sentence in the SemEval training and dev sets to "*subject* and *object*", where *subject* and *object* are the actual entities in the sentence. Surprisingly, a trained PA-LSTM model on this data is able to achieve 65.1 F_1 on the dev set if GloVe vectors are used to initialize word vectors, and 47.9 dev F_1 even without GloVe initialization. To further evaluate the model in a more realistic setting, we trained one model with the original SemEval training set (unmasked) and one with entities masked in the training set, following what we have done for TACRED (masked). While the unmasked model achieves a 83.6 F_1 on the original SemEval dev set, performance drops drastically to 62.4 if we replace dev set entities with a special $<UNK>$ token to simulate the presence of unseen entities. In contrast, the masked model is unaffected by unseen entities and achieves a stable dev F_1 of 74.7. This suggests that models trained without entities masked generalize poorly to new examples with unseen entities. Our findings call for more careful evaluation that takes dataset biases into account in future relation extraction studies.

7 Conclusion

We showed the success of a neural architecture based on a graph convolutional network for relation extraction. We also proposed path-centric pruning to improve the robustness of dependency-

[4] We choose the PA-LSTM model because it is more amenable to our experiments with simplified examples.

based models by removing irrelevant content without ignoring crucial information. The combination of these two techniques achieves state-of-the-art performance on the large-scale TACRED dataset.

References

Heike Adel, Benjamin Roth, and Hinrich Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)* .

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)* .

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (EMNLP 2005)*. pages 724–731.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 Interactive poster and demonstration sessions*. Association for Computational Linguistics, page 22.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR 2017)* .

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)* .

Yang Liu, Sujian Li, Furu Wei, and Heng Ji. 2016. Relation classification via modeling augmented dependency paths. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24(9):1585–1594.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)* .

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. pages 1003–1011.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. volume 14, pages 1532–1543.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. *Proceedings of the 15th Conference of the European Association for Computational Linguistics (EACL 2017)* .

Bryan Rink and Sanda Harabagiu. 2010. UTD: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 256–259.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*. pages 4974–4983.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL 2015)* .

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*. pages 1785–1794.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)* .

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research* 3:1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2014)*. pages 2335–2344.

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

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

A Experimental Details

A.1 Hyperparameters

TACRED We set LSTM hidden size to 200 in all neural models. We also use hidden size 200 for the output feedforward layers in the GCN model. We use 2 GCN layers and 2 feedforward (FFNN) layers in our experiments. We employ the ReLU function for all nonlinearities in the GCN layers and the standard max pooling operations in all pooling layers. For the Tree-LSTM model, we find a 2-layer architecture works substantially better than the vanilla 1-layer model, and use it in all our experiments. For both the Tree-LSTM and our models, we apply path-centric pruning with $K = 1$, as we find that this generates best results for all models (also see Figure 3). We use the pre-trained 300-dimensional GloVe vectors (Pennington et al., 2014) to initialize word embeddings, and we use embedding size of 30 for all other embeddings (i.e., POS, NER). We use the dependency parse trees, POS and NER sequences as included in the original release of the dataset. For regularization we apply dropout with $p = 0.5$ to all LSTM layers and all but the last GCN layers.

SemEval We use LSTM hidden size of 100 and use 1 GCN layer for the SemEval dataset. We preprocess the dataset with Stanford CoreNLP (Manning et al., 2014) to generate the dependency parse trees, POS and NER annotations. All other hyperparameters are set to be the same.

A.2 Training

For training we use Stochastic Gradient Descent with initial learning rate of 1.0. We use a cutoff of 5 for gradient clipping. For GCN models, we train every model for 100 epochs on the TACRED dataset, and from epoch 5 we start to anneal the learning rate by a factor of 0.9 every time the F_1 score on the dev set does not increase after an epoch. For Tree-LSTM models we find 30 total epochs to be enough. Due to the small size of the SemEval dataset, we train all models for 150 epochs, and use an initial learning rate of 0.5 with a decay rate of 0.95.

B Comparing GCN models and PA-LSTM on TACRED

We compared the performance of both GCN models with the PA-LSTM on the TACRED dev set. To minimize randomness that is not inherent to these models, we accumulate statistics over 5 independent runs of each model, and report them in Figure 6. As is shown in the figure, both GCN models capture very different examples from the PA-LSTM model. In the entire dev set of 22,631 examples, 1,349 had at least 3 more GCN models predicting the label correctly compared to the PA-LSTM, and 1,545 saw an improvement from using the PA-LSTM. The C-GCN, on the other hand, outperformed the PA-LSTM by at least 3 models on a total of 906 examples, and lost by a margin of at least 3 on another 838 examples, as reported in the main text. This smaller difference is also reflected in the diminished gain from ensembling with the PA-LSTM shown in Table 1. We hypothesize that the diminishing difference results from the LSTM contextualization layer, which incorporates more information readily available at the surface form, rendering the model's behavior more similar to a sequence model.

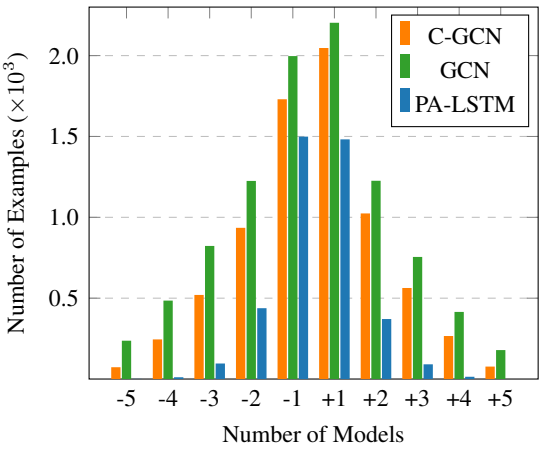

Figure 6: Aggregated 5-run difference compared to PA-LSTM on the TACRED dev set. For each example, if X out of 5 GCN models predicted its label correctly and Y PA-LSTM models did, it is aggregated in the bar labeled $X-Y$. "0" is omitted from the plot due to redundancy.

For reference, we also include in Figure 6 the comparison of another 5 different runs (with different seeds) of the PA-LSTM to the original 5 runs of the PA-LSTM. This is to confirm that the difference shown in the figure between the model classes is indeed due a to model difference, rather than an effect of different random seeds. More specifically, the two groups of PA-LSTM only see 108 and 105 examples exceeding the 3-model margin on either side over the 5 runs, much lower than the numbers reported above for the GCN models.