Relation Extraction Exploiting Full Dependency Forests

Lifeng Jin,[1*†] Linfeng Song,[2†] Yue Zhang,[3,4] Kun Xu,[2] Wei-yun Ma,[5*] Dong Yu[2]
[1]The Ohio State University, Columbus, OH, USA
[2]Tencent AI Lab, Bellevue, WA, USA
[3]Institute of Advanced Technology, Westlake Institute for Advanced Study
[4]School of Engineering, Westlake Universtiy
[5]Academia Sinica

Abstract

Dependency syntax has long been recognized as a crucial source of features for relation extraction. Previous work considers 1-best trees produced by a parser during preprocessing. However, error propagation from the out-of-domain parser may impact the relation extraction performance. We propose to leverage full dependency forests for this task, where a full dependency forest encodes all possible trees. Such representations of full dependency forests provide a differentiable connection between a parser and a relation extraction model, and thus we are also able to study adjusting the parser parameters based on end-task loss. Experiments on three datasets show that full dependency forests and parser adjustment give significant improvements over carefully designed baselines, showing state-of-the-art or competitive performances on biomedical or newswire benchmarks.

1 Introduction

As a central task in automatically extracting information, relation extraction (Chinchor, 1998) aims to determine the relation between a pair of entity mentions. It has been shown to be useful to general-purpose natural language understanding and other downstream tasks, such as knowledge-base completion (Surdeanu et al., 2012; Riedel et al., 2013) and KBQA (Yih et al., 2015; Xu et al., 2016; Yu et al., 2017). In the biomedical domain, it can help doctors make accurate decisions by mining supportive or contradictory evidences from recently published research articles (Quirk and Poon, 2017; Peng et al., 2017). This is important as there are thousands of new medical articles released everyday, making it impossible to track them manually.

Syntactic features have been shown useful for relation extraction. Early work utilized surface features (Mooney and Bunescu, 2006) or shallow syntactic information (e.g. base-NP chunks) (Zhou et al., 2005). Subsequent research (Culotta and Sorensen, 2004; Bunescu and Mooney, 2005; Jiang and Zhai, 2007; Zhou et al., 2007; Nguyen, Moschitti, and Riccardi, 2009) suggested using syntax trees for better capturing long-range dependencies (e.g. via kernel functions).

[*]Work done when L. Jin and W. Ma were at Tencent AI Lab.
[†]Equal contribution

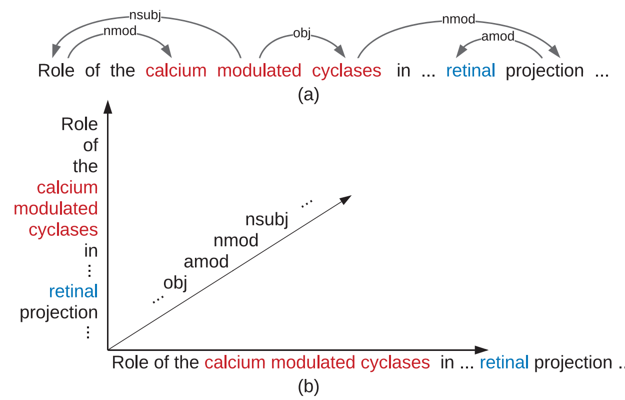

Figure 1: (a) 1-best dependency tree and (b) 3-dimensional full dependency forest for a biomedical sentence. Associated mentions are in different colors. Phrase “calcium modulated cyclases” are broken in the 1-best tree, because “modulated” and “calcium” are erroneously treated as the sentence main verb and a modifier of the main subject, respectively.

Recent work (Xu et al., 2015; Liu et al., 2015; Miwa and Bansal, 2016; Zhang, Qi, and Manning, 2018; Fu, Li, and Ma, 2019) adopting neural models also saw improvements when incorporating syntactic information into the models.

In addition to the news domain, syntax is also crucial for relation extraction in biomedical domain (Quirk and Poon, 2017; Peng et al., 2017; Song et al., 2018), and one important reason is that biomedical sentences usually contain more domain-specific words (e.g. new medicines, molecules and gene IDs) than newswire text. These words introduce more data sparsity, hindering the effectiveness of surface features. Compared with sequential surface-level structures, syntax helps to model word-to-word relations by drawing direct connections between distant yet syntactically related words, and thus capturing informative structural knowledge than surface features.

Existing syntax-based relation extraction models, however, can suffer from two major issues. First, they take the 1-best trees generated during preprocessing, and thus error propagation is introduced because of parsing mistakes. This can be much severer for the biomedical domain, where parsing accuracies can be significantly worse than the news domain (McClosky and Charniak, 2008; Candito, Anguiano, and Seddah, 2011). Figure 1(a) shows the 1-best dependency tree of a biomedical sentence in our dataset, where the dependency tree contains a serious mistake. In particular, the phrase "calcium modulated cyclases" is broken because "modulated" is mistakenly considered as the main verb of the whole sentence, and "calcium" is predicated as the modifier of the main subject in the sentence. Second, 1-best trees are discrete structures, which adds an additional layer of difficulties if we want to finetune the parser parameters during relation extraction training (Yogatama et al., 2017).

In this paper, we tackle these two issues above by leveraging full dependency forests for relation extraction. As shown in Figure 1(b), we define a full forest as a 3-dimensional tensor, with each point representing the conditional probability $p(w_j, l|w_i)$ of one word w_i modifying another word w_j with a relation l. Compared with a 1-best tree, a full dependency forest efficiently represents all possible dependency trees within a compact and dense structure, containing all possible trees (including the gold tree).

We directly mine useful knowledge from each full forest using a convolutional neural network (CNN) (LeCun, Bengio, and others, 1995). CNNs have shown to be effective on handling dense multi-dimensional data, such as images and videos. In order to allow our model to learn useful features associated with the target mentions during encoding, we parameterize our convolutional kernels with the target mention pairs. Similar parameterization of convolutional kernels was recently studied on aspect-based sentiment analysis (Huang and Carley, 2018) and demonstrated positive effects.

Results on two relation extraction benchmarks in biomedical domain and one benchmark in general news domain show that our method outperforms the best previous numbers and strong baselines that use 1-best dependency trees as features, achieving the state-of-the-art or competitive performances in the literature. To our knowledge, we are the first to study full dependency forests and task-oriented parser adjustment for relation extraction, showing their advantages over using 1-best discrete trees. Our code is now available at https://github.com/lifengjin/xxx.

2 Task definition

Formally, the input to our task is a sentence $\boldsymbol{s} = w_1, w_2, \ldots, w_N$, where N is the number of words in the sentence and w_i is the i-th input word. $\boldsymbol{s}$ is annotated with the boundary information ($\xi_1 : \xi_2$ and $\zeta_1 : \zeta_2$) of associated mentions (ξ and ζ). We focus on the binary relation extraction setting (Quirk and Poon, 2017), where the number of associated mentions is two. The output is a relation from a predefined relation set $\boldsymbol{R} = (r_1, \ldots, r_M, \text{None})$, where "None" means that no relation holds for the mentions.

3 Baseline: using 1-best trees

Our baseline model stacks a bidirectional LSTM layer to encode an input sentence with a graph recurrent network (GRN) (Song et al., 2018; Beck, Haffari, and Cohn, 2018; Zhang, Liu, and Song, 2018) to encode a 1-best dependency tree. They extract features from the sentence and the dependency tree, respectively. Similar model frameworks (Zhang, Qi, and Manning, 2018; Song et al., 2018; Zhu et al., 2019) have recently shown highly competitive performances for relation extraction using 1-best trees.

3.1 Bi-LSTM layer

Given an input sentence $w_1, w_2, \ldots, w_N$, we first represent the input words with their embeddings $\boldsymbol{e}_1, \boldsymbol{e}_2 \ldots \boldsymbol{e}_N$.[1] A Bi-LSTM layer is used to encode the sentences:

$$\begin{aligned} \overleftarrow{\boldsymbol{h}}_i^{(0)} &= \text{LSTM}_l(\overleftarrow{\boldsymbol{h}}_{i+1}^{(0)}, \boldsymbol{e}_i) \\ \overrightarrow{\boldsymbol{h}}_i^{(0)} &= \text{LSTM}_r(\overrightarrow{\boldsymbol{h}}_{i-1}^{(0)}, \boldsymbol{e}_i), \end{aligned} \tag{1}$$

The state of each word w_i is generated by concatenating the states in both directions:

$$\boldsymbol{h}_i^{(0)} = [\overleftarrow{\boldsymbol{h}}_i^{(0)}; \overrightarrow{\boldsymbol{h}}_i^{(0)}] \tag{2}$$

3.2 GRN layer

As a next step, a 1-best dependency tree from a dependency parser is organized as a directed graph $\boldsymbol{D}_T = \langle \boldsymbol{V}, \boldsymbol{E} \rangle$, where $\boldsymbol{V}$ includes all words $w_1, w_2, \ldots, w_N$ and $\boldsymbol{E} = \{(w_j, l, w_i)\}|_{w_j \in V, w_i \in V}$ represents all dependency edges. Each triple (w_j, l, w_i) corresponds to a dependency edge, where w_j modifies w_i with an arc label $l \in L$. Each word w_i is associated with a hidden state, which is initialized with the Bi-LSTM output $\boldsymbol{h}_i^{(0)}$. The state representation of the entire tree includes all word states:

$$\boldsymbol{h}^{(0)} = \{\boldsymbol{h}_i^{(0)}\}_{w_i \in V} \tag{3}$$

To capture non-local information over the tree, GRN adopts an iterative message passing framework, performing information exchange between directly connected words in each iteration. As a result, each word state is updated by absorbing a larger context through the process, and a sequence of states $\boldsymbol{h}^{(0)}, \boldsymbol{h}^{(1)}, \ldots$ is generated for the entire tree. The final state $\boldsymbol{h}^{(T)} = \text{GRN}(\boldsymbol{h}^{(0)}, T)$ is used to represent the dependency tree. Below we give more details:

Message passing The message passing framework takes two main steps within each iteration: message calculation and state update. Take w_i at iteration t for example. In the first step, two messages $\boldsymbol{m}_i^{\uparrow}$ and $\boldsymbol{m}_i^{\downarrow}$ are calculated by summing up the messages from its children and parent in the dependency tree, respectively, and a message from a child/parent consists of its hidden states concatenated with the corresponding edge label embedding:

$$\begin{aligned} \boldsymbol{m}_i^{\uparrow} &= \sum_{(w_j, l, w_i) \in \boldsymbol{E}_{(\cdot,\cdot,i)}} [\boldsymbol{h}_j^{(t-1)}; \boldsymbol{e}_l] \\ \boldsymbol{m}_i^{\downarrow} &= \sum_{(w_i, l, w_k) \in \boldsymbol{E}_{(i,\cdot,\cdot)}} [\boldsymbol{h}_k^{(t-1)}; \boldsymbol{e}_{l_{rev}}], \end{aligned} \tag{4}$$

[1]For model variables, lowercase italic letters are for scalars and indices, lowercase bold letters are for vectors, uppercase letters are for matrices and uppercase bold letters are for higher order tensors.

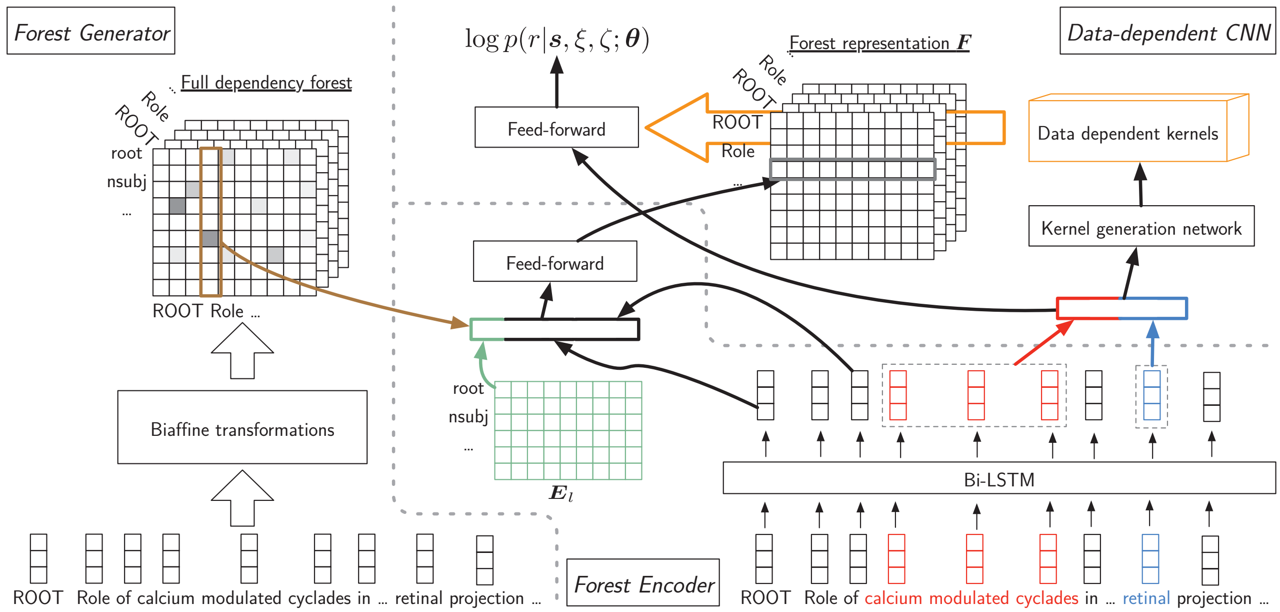

Figure 2: Model framework containing a *forest generator* and a *forest encoder* based on data dependent CNN.

where $\boldsymbol{E}_{(\cdot,\cdot,i)}$ and $\boldsymbol{E}_{(i,\cdot,\cdot)}$ represent the edges connected with w_i as head and modifier, respectively, and $\boldsymbol{e}_x$ represents the embedding of label x. l_{rev} is the reversed version of original label l (e.g. "amod-rev" is the reversed version of "amod").

Next, GRN adopts standard LSTM operations (Hochreiter and Schmidhuber, 1997) to update its hidden state $\boldsymbol{h}_i^{(t-1)}$ with the integrated message, taking a cell $\boldsymbol{c}_i^{(t)}$ to record memory for $\boldsymbol{h}_i^{(t)}$:

$$\boldsymbol{h}_i^{(t)}, \boldsymbol{c}_i^{(t)} = \text{LSTM}([\boldsymbol{m}_i^{\uparrow}; \boldsymbol{m}_i^{\downarrow}], \boldsymbol{c}_i^{(t-1)}), \tag{5}$$

where cell vector $\boldsymbol{c}_i^{(0)}$ is initialized as a vector of zeros.

The same process repeats for T iterations. Starting from $\boldsymbol{h}^{(0)}$ of the Bi-LSTM layer, increasingly informed hidden states $\boldsymbol{h}^{(t)}$ are obtained with increasing iterations and $\boldsymbol{h}^{(T)}$ is used as the final representation of each word.

3.3 Relation prediction

Taking the outputs $\boldsymbol{h}^{(T)}$ of the GRN layer, we calculate the representation vectors of the two target mentions ξ and ζ by averaging among their tokens:

$$\boldsymbol{h}_\xi = f_{\text{avg}}(\boldsymbol{h}^{(T)}_{\xi_1:\xi_2}); \quad \boldsymbol{h}_\zeta = f_{\text{avg}}(\boldsymbol{h}^{(T)}_{\zeta_1:\zeta_2}) \tag{6}$$

where $\xi_1 : \xi_2$ and $\zeta_1 : \zeta_2$ represent the spans of ξ and ζ, respectively, and f_{avg} is the averaging function. Finally, the representations of both mentions are concatenated to be the input of a logistic regression classifier:

$$\boldsymbol{y} = \text{softmax}(W[\boldsymbol{h}_\xi; \boldsymbol{h}_\zeta] + \boldsymbol{b}), \tag{7}$$

where W and $\boldsymbol{b}$ are model parameters.

4 Model: using full dependency forests

A full dependency forest is a fully connected graph, where the vertices are input words and each edge represents a dependency relation l (such as "subj" and "nmod") between two words. An edge is also associated with a weight that represents the conditional probability of the relation $p(w_j, l|w_i)$. Compared with 1-best trees, full dependency forests encode all possible dependency relations with their parser confidence scores. In order to efficiently encode them, we represent each forest as a 3-dimensional tensor of probabilities, with the first two dimensions corresponding to input words and the remaining dimension corresponding to dependency arcs, as shown in Figure 1(b).

Our model is similar to the baseline by stacking a Bi-LSTM layer to process each input sentence with a network based on convolutional operations to handle full dependency forests. We use a first-order graph-based dependency parser (Dozat and Manning, 2017) for generating full dependency forests (shown in the left part of Figure 2). The Bi-LSTM layer (shown in the right bottom corner of Figure 2) is identical with our baseline as described in Section 3.1.

4.1 Forest generation

As shown in Figure 2, the forest generator takes the neural network architecture of the deep biaffine parser (Dozat and Manning, 2017), where input words $w_1, \ldots, w_N$ are first represented by their embeddings, before being processed by a Bi-LSTM layer to obtain their contextual representations $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N$. As a next step, the representations for a word w_i being the head or the dependent of any dependency relation are calculated by passing its contextual vector $\boldsymbol{r}_i$ through the corresponding multi-layer perceptrons (MLPs):

$$\boldsymbol{h}_i^{\text{dep}} = \text{MLP}_1^{\text{dep}}(\boldsymbol{r}_i); \quad \boldsymbol{h}_i^{\text{head}} = \text{MLP}_1^{\text{head}}(\boldsymbol{r}_i) \tag{8}$$

Then the scores for all relation labels given a head word w_j and a dependent word w_i are calculated as:

$$\boldsymbol{s}_{i,j}^{\text{label}} = \boldsymbol{h}_j^{\text{head}} \boldsymbol{U_l} \boldsymbol{h}_i^{\text{dep}} + (\boldsymbol{h}_j^{\text{head}} \oplus \boldsymbol{h}_i^{\text{dep}})^\top V_l + \boldsymbol{b}_l, \quad (9)$$

and the scores for each unlabeled arc with any possible head word given a dependent word w_i are calculated as:

$$\hat{\boldsymbol{h}}_i^{\text{dep}} = \text{MLP}_{\text{u}}^{\text{dep}}(\boldsymbol{r}_i); \;\; \hat{\boldsymbol{h}}_i^{\text{head}} = \text{MLP}_{\text{u}}^{\text{head}}(\boldsymbol{r}_i) \quad (10)$$

$$\boldsymbol{s}_i^{\text{head}} = \hat{H}^{\text{head}} U_a \hat{\boldsymbol{h}}_i^{\text{dep}} + \hat{H}^{\text{head}} \boldsymbol{v}_a, \quad (11)$$

where $\boldsymbol{U_l}, V_l, \boldsymbol{b}_l, U_a$ and $\boldsymbol{v}_a$ are all model parameters. Finally the conditional probability of each label l and each head word w_j given a dependent word w_i is calculated as:

$$\begin{aligned} p(w_j, l|w_i) &= p(l|w_j, w_i) \times p(w_j|w_i) \\ &= \text{softmax}(\boldsymbol{s}_{i,j}^{\text{label}})_{(l)} \times \text{softmax}(\boldsymbol{s}_i^{\text{head}})_{(j)}, \end{aligned} \quad (12)$$

where (x) in the subscript represents choosing the x-th item from the corresponding vector.

Given the words of an input sentence, the probabilities from Equation 12 can be organized into a rank-3 tensor. This exactly fits our forest definition, so that no further modification is required before the probabilities are processed by our model. For the baseline, we apply a minimum spanning tree algorithm over the outputs to generate the 1-best tree, which is then encoded by a GRN described in Section 3.2.

4.2 Forest representation

The generated forests only contain the probabilities of word-to-word relations without any lexical or relational knowledge, thus it is necessary to integrate these probabilities with word and relation embeddings. The middle part of Figure 2 visualizes our approach for the integration. In particular, for each word pair relation (w_j, l, w_i), the word hidden states ($\boldsymbol{h}_j^{(0)}$ and $\boldsymbol{h}_i^{(0)}$, produced by Equation 2) and relation embedding ($\boldsymbol{e}_l \in \boldsymbol{E}_l$) are first concatenated, and then the possible relations are marginalized by a weighted sum:

$$\boldsymbol{h}_{w_j,w_i} = \sum_{l \in L} p(w_j, l|w_i) [\, \boldsymbol{h}_j^{(0)}; \boldsymbol{h}_i^{(0)}; \boldsymbol{e}_l \,] \quad (13)$$

By using a weighted sum, relations with high probabilities are highlighted, while other relations are also preserved. As a next step, a linear layer is applied on the results of the weighted sum to correlate the concatenated word states and relation embedding:

$$\boldsymbol{h}'_{w_j,w_i} = W_f \boldsymbol{h}_{w_j,w_i} + \boldsymbol{b}_f \quad (14)$$

where W_f and $\boldsymbol{b}_f$ are trainable parameters. The resulting representations are organized following the word order of the original sentence into a rank-3 tensor $\boldsymbol{F} \in \mathbb{R}^{|s| \times |s| \times d}$, where $|s|$ is the length of the sentence[2] and d is the length of the representation of a word pair ($\boldsymbol{h}'_{w_j,w_i}$).

[2] With the special symbol ROOT considered as the first word.

4.3 Feature extraction with data-dependent CNN

As shown on the top right of Figure 2, each generated forest representation is a 3-dimensional tensor ($\boldsymbol{F}$), with each lowest-rank vector corresponding to the relation and content of a word pair. We choose to apply convolutional operations to $\boldsymbol{F}$, which have been shown effective in handling dense multi-dimensional data, to extract useful features from this structure.

For *conventional* CNNs, the convolution kernels $\boldsymbol{W}_c \in \mathbb{R}^{n_f \times n_w \times n_h \times d}$ are first randomly initialized, where n_f, n_w and n_h are the kernel count, kernel width and height, respectively. As the next step, the kernels are applied to each convolutional region $\tilde{\boldsymbol{F}}_{(i,j)} = \boldsymbol{F}_{(i:i+n_w; j:j+n_h; \cdot)}$ with the coordinate of its upper left element being $(i, j, 0)$, and the output of the convolution is:

$$\boldsymbol{C}_{(\cdot,i,j)} = \text{ReLU}(\boldsymbol{W}_c \tilde{\boldsymbol{F}}_{(i,j)} + \boldsymbol{b}_c), \quad (15)$$

where $\boldsymbol{b}_c$ is trainable parameter representing kernel bias. On top of the convolution outputs, max pooling is then applied to reduce $\boldsymbol{C}$ into a vector $\hat{\boldsymbol{c}} \in \mathbb{R}^{n_f}$ by keeping the maximal value for each filter: $\hat{\boldsymbol{c}} = f_{\max}(\boldsymbol{C})$, and $\hat{\boldsymbol{c}}$ will be part of the input to our final relation predictor (Section 4.4).

One major drawback of conventional CNNs for relation extraction is that their kernels only depend on the input sentence, and thus the extracted features will be the same for the same sentence. However, for relation extraction, one sentence can contain more than two mentions and the relations for different pairs of mentions in the same sentence can be entirely different. In order for our feature extraction procedure to be aware of the target mentions that are critical for predicting the correct relation, we introduce *data-dependent CNN*, where a kernel generation network is adopted to produce data-dependent convolution kernels (the orange 3D box in Figure 2) by taking mention representations as inputs.

Data-dependent CNN For data-dependent CNN, the convolution kernels $\boldsymbol{W}_c$ are produced from our kernel generation network instead of randomly initialized. Taking the representation vectors of both target mentions ($\boldsymbol{h}_\xi$ and $\boldsymbol{h}_\zeta$ as shown in Equation 6), the kernel generation network first correlates them with a fully-connected layer:

$$\boldsymbol{h}_{\xi,\zeta} = \text{ReLU}(W_d[\boldsymbol{h}_\xi; \boldsymbol{h}_\zeta] + \boldsymbol{b}_d), \quad (16)$$

before a set of data-dependent convolution kernels being calculated with another multi-dimensional layer:

$$\boldsymbol{W}_c = \text{ReLU}(\boldsymbol{W}_p \boldsymbol{h}_{\xi,\zeta} + \boldsymbol{B}_p), \quad (17)$$

where $W_d, \boldsymbol{b}_d, \boldsymbol{W}_p, \boldsymbol{B}_p$ are model parameters. Next, the generated kernels are applied on $\boldsymbol{F}$ to extract useful features, which is identical with Equation 15, before max pooling being used to calculate $\hat{\boldsymbol{c}}$.

4.4 Relation prediction

For relation prediction, the extracted features ($\hat{\boldsymbol{c}}$) are combined with the mention representations and feed into a logistic regression classifier:

$$\begin{aligned} \boldsymbol{h}'_{\xi,\zeta} = \; & \text{ReLU}(W_{g_1}[\boldsymbol{h}_\xi; \boldsymbol{h}_\zeta] + \boldsymbol{b}_{g_1}) + \\ & \text{ReLU}(W_{g_2}\hat{\boldsymbol{c}} + \boldsymbol{b}_{g_2}), \end{aligned} \quad (18)$$

$$\boldsymbol{y} = \text{softmax}(W_r \boldsymbol{h}'_{\xi,\zeta} + \boldsymbol{b}_r), \quad (19)$$

where $W_{g_1}, \boldsymbol{b}_{g_1}, W_{g_2}, \boldsymbol{b}_{g_2}, W_r$ and $\boldsymbol{b}_r$ are model parameters.

4.5 Comparison with the baseline

There are two differences between our model and baseline: one is the syntactic features being used (trees vs forests), the other is the network to encode such features (GRN vs DDCNN). Empirically, GRN and DDCNN give the best results for their respective syntax representations, namely trees and full forests. In our experiments, we introduce more baselines to pinpoint the specific gains regarding the syntax and the model structure, respectively.

5 Training

We train the baseline and our model using cross-entropy loss. For each training instance that contains a sentence $\boldsymbol{s}$ with two target mentions ξ and ζ, the cross-entropy loss between the gold relation r and model distribution is:

$$l = -\log p(r|\boldsymbol{s}, \xi, \zeta; \boldsymbol{\theta}), \tag{20}$$

where $\boldsymbol{\theta}$ represents the model parameters. For models using parser adjustment, $\boldsymbol{\theta}$ includes the parameters of the forest generator in addition to other model components.

6 Experiments

We conduct thorough comparisons between our model leveraging full dependency forests, 1-best trees and only surface strings on the relation extraction task described in Section 2.

6.1 Data

Our main goal is to perform biomedical relation extraction, as the parsing errors are much severer in the biomedical domain than in the news domain. We choose two standard benchmarks in the biomedical domain, where parsing performance drops dramatically due to domain variance and unknown words. In addition, we conduct evaluation on SemEval-2010 task 8 (Hendrickx et al., 2009), a benchmark dataset for relation extraction in the news domain.

BioCreative VI CPR (Krallinger et al., 2017) This task focuses on the relations between chemical compounds (such as drugs) and proteins (such as genes). The corpus contains 1020, 612 and 800 extracted PubMed[3] abstracts for training, development and testing, respectively. All abstracts are manually annotated with the mention boundaries and their relations. The data provides three types of NEs: “CHEMICAL”, “GENE-Y” and “GENE-N”, and the relation set $\boldsymbol{R}$ contains 5 regular relations (“CPR:3”, “CPR:4”, “CPR:5”, “CPR:6” and “CPR:9”) and the “None” relation. We segment each abstract into sentences, keeping only the sentences that contain at least a chemical mention and a protein mention. As a result, we obtain 16,107 training, 10,030 development and 14,269 testing instances, in which around 23% have regular relations. By doing this, we effectively sacrifice cross-sentence relations (which are rare) by treating their relation as “None”. This is necessary for efficient generation of dependency structures since directly parsing a short paragraph is slow and introduces more errors. We report F1 scores of the full test set for a fair comparison, using all gold regular relations to calculate recalls.

[3]https://www.ncbi.nlm.nih.gov/pubmed/

Phenotype-Gene relation (PGR) (Sousa, Lamurias, and Couto, 2019) This dataset concerns the relations between human phenotypes (such as diseases) with human gene, where the relation set is a *binary* class on whether a phenotype is related to a gene. It has 18,451 silver training instances and 220 high-quality test instances, with each containing mention boundary annotations. We separate the first 15% training instances as our development set.

SemEval-2010 task 8 (Hendrickx et al., 2009) The SemEval dataset is a widely used benchmark in recent work for general-domain relation extraction. It is a public dataset, which contains 10,717 instances (8000 for training) with 9 relations (such as “Cause-Effect” and “Content-Container”) and a special “None” class.

6.2 Models

To study the effectiveness of forests, we compare our method with the following baselines:

- TREE-GRN: It corresponds to our baseline in Section 3, which encodes 1-best trees with a GRN.
- TREE-DDCNN: It is similar to TREE-GRN except that the 1-best trees are organized as sparse 3D tensors and are encoded with the DDCNN model. This is for calibrating the contribution of using forests.
- EQUAL-DDCNN: It represents the baseline without any syntactic information. Taking the DDCNN in Section 4, it consumes dense forests with equivalent edge weights instead of parser generated probabilities.
- RANDOMFT-DDCNN: It adopts our model in Section 4 with parser adjustment based on the relation extraction loss, except that the parser is initialized randomly.
- FOREST-CNN: It takes the model in Section 4, but using a regular CNN to consume forests.

We further compare two versions of our model *with or without finetuning*:

- FOREST-DDCNN: It represents our model in Section 4 *without* parser adjustment for relation extraction training.
- FORESTFT-DDCNN: It shows our model in Section 4 *with* parser adjustment. This model takes the *same* number of parameters as RANDOMFT-DDCNN.

6.3 Settings

We pretrain our parser (Dozat and Manning, 2017) on the Penn Treebank (PTB) (Marcus and Marcinkiewicz, 1993) converted to Stanford Dependency v3.5 to obtain 1-best trees and full dependency forests. Using standard PTB data split (section 02–21 for training, 22 for development and 23 for testing), the parser gives UAS and LAS scores of 95.7 and 94.6, respectively.

For biomedical experiments, word embeddings of our baseline and model are initialized with the 200-dimensional BioASQ vectors[4] pretrained on 10M abstracts of biomedical articles, while we use 300-dimensional Glove embeddings[5]

[4]http://bioasq.lip6.fr/tools/BioASQword2vec/

[5]https://nlp.stanford.edu/projects/glove/

Syntax type	Model	test F1
None	GRU+Attn (Liu et al., 2017)†	49.5
	Bran (Verga et al., 2018)†	50.8
	EQUAL-DDCNN	50.4
	RANDOMFT-DDCNN	45.4
Tree	TREE-GRN	51.4
	TREE-DDCNN	50.3
Forest	Forest-GRN (Song et al., 2019)†	53.4
	FOREST-CNN	50.5
	FOREST-DDCNN	53.1
	FORESTFT-DDCNN	**55.7**

Table 1: Main results on BioCreative VI CPR. † denotes previous numbers. We use the same notation for later results.

pretrained on 840B data for the news-domain experiments. These embeddings are fixed during relation extraction training. The dimension of hidden vectors in Bi-LSTM is set to 200, and the number of filters is also set to 200. We use Adam (Kingma and Ba, 2014), with a learning rate of 0.001, as the optimizer. For baseline, GRN step T is set to 2. These hyperparameter values are chosen based on either report by existing work or development experiments.

6.4 Main results

Table 1 compares our models with our baselines and previously reported state-of-the-art numbers, where systems using the same type of syntax forms are grouped together. Among previous work, *GRU+Attn* (Liu et al., 2017) stacks a self-attention layer on top of GRU and embedding layers; *Bran* (Verga et al., 2018) adopts a biaffine self-attention model to simultaneously extract the relations of all mention pairs. Both methods use only textual knowledge. As a very recent work, *Forest-GRN* first generates forests as discrete structures by pruning out low-quality edges, before consuming these forests with a GRN network.

With 1-best dependency trees, the TREE-GRN baseline gives a slightly better performance over the previous best system with only textual features (*Bran*), showing the usefulness of dependency information. Using full dependency forests, FOREST-DDCNN gives a large improvement of 1.7 absolute F1 points over TREE-GRN. With parser adjustment, FORESTFT-DDCNN demonstrates a further performance boost of 2.6 absolute points over FOREST-DDCNN, demonstrating the usefulness of task-oriented parser finetuning. Also, FORESTFT-DDCNN outperforms *Forest-GRN*, the previous state-of-the-art model. FOREST-CNN is much worse than FOREST-DDCNN, indicating the importance of letting CNN be aware of the target mentions. In addition to the types of syntax being used (tree vs forest), the encoders are also different (GRN vs DDCNN). We make additional comparison between other baselines and our models to study each factor and lead to the following conclusions:

Effectiveness of full dependency forests TREE-DDCNN shows a lower F1 score than TREE-GRN, while FOREST-DDCNN is much better than TREE-GRN. Both results indicate that the gain of FOREST-DDCNN comes from using forest, not from adopting a different encoder. The lower performance of TREE-DDCNN than TREE-GRN is likely because CNNs are good at handling dense vectors, while trees are highly sparse.

Tree	UAS/LAS	Model	test F1
1K	88.07/85.48	TREE-GRN	47.6
		FORESTFT-DDCNN	51.5
5K	92.63/90.77	TREE-GRN	50.3
		FORESTFT-DDCNN	53.7

Table 2: Test results when less trees are available. The full treebank has 39.8K trees for training.

Effectiveness of parser pretraining Being in the same group, neither EQUAL-DDCNN nor RANDOMFT-DDCNN uses a treebank to pretrain their parser. Instead, EQUAL-DDCNN ignores parser outputs and takes 3D forests of identical probabilities, which is equivalent to not using any syntactic information. On the other hand, RANDOMFT-DDCNN relies only on the relation extraction loss to tune its randomly initialized parser parameters. EQUAL-DDCNN shows a much better performance than RANDOMFT-DDCNN, and RANDOMFT-DDCNN suffers from large training variance. The reason can be that relation extraction loss is not sufficient for adjusting the parser parameters well. On the other hand, both FOREST-DDCNN and FORESTFT-DDCNN in the last group show very strong results. In particular, the only difference between RANDOMFT-DDCNN and FORESTFT-DDCNN is on whether loading the graph structure with syntax. Based on all the evidence, we can draw the conclusion that syntax is useful for relation extraction.

Effectiveness of parser finetuning Again, by contrasting the models in the first group (EQUAL-DDCNN vs RANDOMFT-DDCNN) and the models in the last group (FOREST-DDCNN vs FORESTFT-DDCNN), we can conclude that parser finetuning based on relation extraction loss is helpful when the parser is already at a good initial point by treebank-based pretraining.

6.5 Robustness on parsing accuracy

We have shown in Section 6.4 that a dependency parser trained with a domain-general treebank can produce high-quality forests in a target domain, providing more effective information over 1-best trees for relation extraction. This is based on the assumption that the domain-general treebank is in a descent scale so that the parsing accuracy in the target domain is not very low. As a result, it would be important to evaluate the performances of both tree-based and forest-based models when the domain-general treebank only contains a limited number of trees.

Table 2 shows the performance changes of both TREE-GRN and FORESTFT-DDCNN when only 1K or 5K treebank instances are available for parser pretraining. Taking 1K and 5K gold trees, the performance decrease in terms of LAS are 9.1 and 3.8 points compared with the num-

Model	test F1
BO-LSTM (Lamurias et al., 2019)†	52.3
BioBERT (Lee et al., 2019)†	67.2
TREE-GRN	78.9
FORESTFT-DDCNN	**89.3**

Table 3: Relation classification results on PGR.

Model	test F1
C-GCN (Zhang, Qi, and Manning, 2018)†	84.8
C-AGGCN (Guo, Zhang, and Lu, 2019)†	85.7
TREE-GRN	84.6
FORESTFT-DDCNN	85.5

Table 4: Results on SemEval-2010 task 8.

ber with full treebank. Using 1K trees, the performances of both models drop significantly due to the greater parsing noise, yet *ForestFT-DDCNN* still manages to achieve a slightly better number than TREE-GRN (51.4 test F1) taking 40 times of trees (full treebank). This shows the superiority of our model, especially given the situation that 1K trees are available for many languages. Using 5K trees, *ForestFT-DDCNN* manages to be 3.4-point better than TREE-GRN with the same number of trees. Overall, these results indicate that *ForestFT-DDCNN* is more robust to parsing noises than TREE-GRN.

6.6 Results on PGR

Table 3 shows the comparison with previous work on the PGR testset, where our models are significantly better than the existing models. This is likely because the previous models do not utilize all the information from inputs: *BO-LSTM* takes only the words (without arc labels) along the shortest dependency path between the target mentions; the pretrained parameters of *BioBERT* are kept static during the training of relation extraction.

Using 1-best dependency trees, TREE-GRN is better than *BioBERT* by a large margin (11.7 points), confirming the usefulness of syntactic structures. Utilizing full dependency forests, FORESTFT-DDCNN gives another boost of 10.0+ absolute points from TREE-GRN, showing the usefulness of full dependency forests for medical relation extraction.

6.7 Results on SemEval-2010 task 8

Our model is general and can be used on relation extraction in the news domain. As shown in Table 4, we conduct a preliminary study on SemEval-2010 task8 (Hendrickx et al., 2009), a benchmark for relation extraction in news domain. While the DEPTREE baseline achieves similar performance as *C-GCN*, it is roughly 1 point worse than *C-AGGCN*, and one potential reason is that *C-AGGCN* takes more parameters. Using full forests and parser adjustment, FORESTFT-DDCNN outperforms DEPTREE by almost 1 point and is comparable with *C-AGGCN*. The gain by using full forest is less than those in biomedical domain benchmarks, and the reason can be that the parsing performance for newswire is much more accurate than the biomedical domain.

7 Related work

The effectiveness of dependency forests are rarely studied with few exceptions: Tu et al. (2010) leveraged dependency forests for statistic machine translation, and very recently Song et al. (2019) investigated dependency forests for medical relation extraction. These previous efforts use forests as sparse and discrete structures by pruning out edges of low parser confidence during preprocessing, while we present full dependency forests in a continuous 3D tensor. With full dependency forests, no parsing information is lost, and the parser can be easily adjusted with the loss from the end task. This superiority was demonstrated by comparing with Song et al. (2019) in our experiments. To our knowledge, we are the first to investigate full dependency forests, and we are the first to study syntactic forests for relation extraction under a strong neural framework.

8 Conclusion

Forests are complex structures that are more difficult to generate and consume than trees. Because of this reason, previous research on relation extraction tend to use 1-best trees that are generated during preprocessing, even if this could cause severe error propagation. We proposed an efficient and effective relation extraction model that leverage full dependency forests, each of which encodes all valid dependency trees into a dense and continuous 3D space. This method allows us to merge a parser into a relation extraction model so that the parser can be jointly updated based on end-task loss. Extensive experiments show the superiority of forests for RE, which significantly outperform all carefully designed baselines based on 1-best trees or surface strings.

We leave studying the effectiveness of full dependencies for relation extraction in other domains (Augenstein et al., 2017) as future work.

References

Augenstein, I.; Das, M.; Riedel, S.; Vikraman, L.; and McCallum, A. 2017. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. In *SemEval-2017*, 546–555.

Beck, D.; Haffari, G.; and Cohn, T. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of ACL*.

Bunescu, R., and Mooney, R. 2005. A shortest path dependency kernel for relation extraction. In *EMNLP*.

Candito, M.; Anguiano, E. H.; and Seddah, D. 2011. A word clustering approach to domain adaptation: Effective parsing of biomedical texts. In *Proceedings of IWPT*.

Chinchor, N. A. 1998. Overview of MUC-7/MET-2. Technical report, Science Applications International Corp. San Diego CA.

Culotta, A., and Sorensen, J. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL*.

Dozat, T., and Manning, C. D. 2017. Deep biaffine attention for neural dependency parsing. *ICLR*.

Fu, T.-J.; Li, P.-H.; and Ma, W.-Y. 2019. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of ACL*, 1409–1418.

Guo, Z.; Zhang, Y.; and Lu, W. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of ACL*.

Hendrickx, I.; Kim, S. N.; Kozareva, Z.; Nakov, P.; Ó Séaghdha, D.; Padó, S.; Pennacchiotti, M.; Romano, L.; and Szpakowicz, S. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SemEval*, 94–99.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Huang, B., and Carley, K. 2018. Parameterized convolutional neural networks for aspect level sentiment classification. In *EMNLP*.

Jiang, J., and Zhai, C. 2007. A systematic exploration of the feature space for relation extraction. In *NAACL*, 113–120.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krallinger, M.; Rabal, O.; Akhondi, S. A.; et al. 2017. Overview of the biocreative vi chemical-protein interaction track. In *Proceedings of VI BioCreative*.

Lamurias, A.; Sousa, D.; Clarke, L. A.; and Couto, F. M. 2019. BO-LSTM: classifying relations via long short-term memory networks along biomedical ontologies. *BMC bioinformatics* 20(1):10.

LeCun, Y.; Bengio, Y.; et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361(10):1995.

Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C. H.; and Kang, J. 2019. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

Liu, Y.; Wei, F.; Li, S.; Ji, H.; Zhou, M.; and WANG, H. 2015. A dependency-based neural network for relation classification. In *Proceedings of ACL-IJCNLP*.

Liu, S.; Shen, F.; Wang, Y.; Rastegar-Mojarad, M.; Elayavilli, R. K.; Chaudhary, V.; and Liu, H. 2017. Attention-based neural networks for chemical protein relation extraction. In *Proceedings ofthe BioCreative VI Workshop*.

Marcus, M. P., and Marcinkiewicz, M. A. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics* 19(2).

McClosky, D., and Charniak, E. 2008. Self-training for biomedical parsing. In *Proceedings of ACL*.

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

Mooney, R. J., and Bunescu, R. C. 2006. Subsequence kernels for relation extraction. In *Proceedings of NIPS*.

Nguyen, T.-V. T.; Moschitti, A.; and Riccardi, G. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *EMNLP*.

Peng, N.; Poon, H.; Quirk, C.; Toutanova, K.; and Yih, W.-t. 2017. Cross-sentence n-ary relation extraction with graph LSTMs. *TACL*.

Quirk, C., and Poon, H. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *EACL*.

Riedel, S.; Yao, L.; McCallum, A.; and Marlin, B. M. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*, 74–84.

Song, L.; Zhang, Y.; Wang, Z.; and Gildea, D. 2018. N-ary relation extraction using graph state LSTM. In *EMNLP*.

Song, L.; Zhang, Y.; Gildea, D.; Yu, M.; Wang, Z.; et al. 2019. Leveraging dependency forest for neural medical relation extraction. In *EMNLP-IJCNLP*.

Sousa, D.; Lamurias, A.; and Couto, F. M. 2019. A silver standard corpus of human phenotype-gene relations. In *Proceddings of NAACL*.

Surdeanu, M.; Tibshirani, J.; Nallapati, R.; and Manning, C. D. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*, 455–465.

Tu, Z.; Liu, Y.; Hwang, Y.-S.; Liu, Q.; and Lin, S. 2010. Dependency forest for statistical machine translation. In *Proceedings of COLING*.

Verga, P.; Strubell, E.; and McCallum, A. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of NAACL*.

Xu, Y.; Mou, L.; Li, G.; Chen, Y.; Peng, H.; and Jin, Z. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*.

Xu, K.; Reddy, S.; Feng, Y.; Huang, S.; and Zhao, D. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of ACL*, 2326–2336.

Yih, W.-t.; Chang, M.-W.; He, X.; and Gao, J. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*, 1321–1331.

Yogatama, D.; Blunsom, P.; Dyer, C.; Grefenstette, E.; and Ling, W. 2017. Learning to compose words into sentences with reinforcement learning. In *Proceedings of ICLR*.

Yu, M.; Yin, W.; Hasan, K. S.; dos Santos, C.; Xiang, B.; and Zhou, B. 2017. Improved neural relation detection for knowledge base question answering. In *ACL*.

Zhang, Y.; Liu, Q.; and Song, L. 2018. Sentence-state lstm for text representation. In *Proceedings of ACL*, 317–327.

Zhang, Y.; Qi, P.; and Manning, C. D. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*.

Zhou, G.; Su, J.; Zhang, J.; and Zhang, M. 2005. Exploring various knowledge in relation extraction. In *ACL*.

Zhou, G.; Zhang, M.; Ji, D.; and Zhu, Q. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of EMNLP*.

Zhu, H.; Lin, Y.; Liu, Z.; Fu, J.; Chua, T.-s.; and Sun, M. 2019. Graph neural networks with generated parameters for relation extraction. *arXiv preprint arXiv:1902.00756*.