Learning Latent Forests for Medical Relation Extraction

Zhijiang Guo\textsuperscript{1*}, Guoshun Nan\textsuperscript{1*}, Wei Lu\textsuperscript{1} and Shay B. Cohen\textsuperscript{2}
\textsuperscript{1}StatNLP Research Group, Singapore University of Technology and Design
\textsuperscript{2}ILCC, School of Informatics, University of Edinburgh

zhijiang Guo@mymail.sutd.edu.sg, guoshun Nan@sutd.edu.sg, luwei@sutd.edu.sg, scohen@inf.ed.ac.uk

Abstract

The goal of medical relation extraction is to detect relations among entities, such as genes, mutations and drugs in medical texts. Dependency tree structures have been proven useful for this task. Existing approaches to such relation extraction leverage off-the-shelf dependency parsers to obtain a syntactic tree or forest for the text. However, for the medical domain, low parsing accuracy may lead to error propagation downstream the relation extraction pipeline. In this work, we propose a novel model which treats the dependency structure as a latent variable and induces it from the unstructured text in an end-to-end fashion. Our model can be understood as composing task-specific dependency forests that capture non-local interactions for better relation extraction. Extensive results on four datasets show that our model is able to significantly outperform state-of-the-art systems without relying on any direct tree supervision or pre-training.

1 Introduction

With a significant growth in the medical literature, researchers in the area are still required to track it mostly manually. This is an opportunity to automate some of this tracking process, and indeed Natural Language Processing (NLP) techniques have been used to automatically extract knowledge from medical articles. Among these techniques, relation extraction plays a significant role as it facilitates the detection of relations among entities in the medical literature [Peng et al., 2017; Song et al., 2019]. For example in Figure 1, the sub-clause “human phenylalanine hydroxylase catalytic domain with bound catechol inhibitors” drawn from the CPR dataset [Krallinger et al., 2017] contains two entities, namely phenylalanine hydroxylase and catechol. There is a “down regulator” relation between these two entities, denoted as “CPR:4”.

Dependence structures are often used for relation extraction as they are able to capture non-local syntactic relations that are only implicit in the surface form alone [Zhang et al., 2018]. In the medical domain, early efforts leverage graph LSTM [Peng et al., 2017] or graph neural networks [Song et al., 2018; Guo et al., 2019a] to encode the 1-best dependency tree. However, dependency parsing accuracy is relatively low in the medical domain. Figure 1 shows the 1-best dependency tree obtained by the Stanford CoreNLP [Manning et al., 2014], where the dependency tree contains an error. In particular, the entity phrase phenylalanine hydroxylase is broken since the word hydroxylase is mistakenly considered as the main verb. In order to mitigate the error propagation when incorporating the dependency structure, Song et al. [2019] construct a dependency forest by adding additional edges with high confidence scores given by a dependency parser trained on the news domain or merging the K-bests trees [Eisner, 1996] by combining identical dependency edges. Lifeng et al. [2020] jointly train a pre-trained dependency parser [Dozat and Manning, 2017] and a relation extraction model. The dependency forest generated by the parser is a 3-dimensional tensor, with each position representing the conditional probability of one word modifying another word with a relation, which encodes all possible dependency relations with the confidence scores.

Unlike previous research efforts that rely on dependency parsers trained on newswire text, our proposed model treats the dependency parse as a latent variable and induces it in an end-to-end fashion. We build our model based on the mechanism of structured attention [Kim et al., 2017;
Liu and Lapata, 2018]. Using a variant of the Matrix-Tree Theorem [Tutte, 1984; Koo et al., 2007], our model is able to generate task-specific non-projective dependency structures for capturing non-local interactions between entities without recourse to any pre-trained parsers or tree supervision. We further construct multiple forests by projecting the representations of nodes to different representation subspaces, allowing an induction of a more informative latent structure for better relation prediction. We name our proposed model as LF-GCN, where LF is the abbreviation of latent forests.

Experiments show that our LF-GCN model is able to achieve better performance on various relation extraction tasks. For the sentence-level tasks, our model surpasses the current stat-of-the-art models on the CPR dataset [Krallinger et al., 2017] and the PGR dataset [Sousa et al., 2019] by 3.2% and 2.6% in terms of F1 score, respectively. For the cross-sentence tasks [Peng et al., 2017], our model is also consistently better than others, showing its effectiveness on long medical text. We release our code at https://github.com/Cartus/Latent-Forests.

2 Model

Here we present the proposed model as shown in Figure 2.

2.1 Forest Inducer

Existing approaches leverage a dependency parser trained on newswire text [Song et al., 2019] or a fine-tuned parser for the medical domain [Lifeng et al., 2020] to generate a dependency forest, which is a fully-connected weighted graph. Unlike previous efforts, we treat the forest as a latent variable, which can be learned from a targeting dataset in an end-to-end manner. Inspired by Kim et al. [2017] and Liu and Lapata [2018], we use a variant of Kirchhoff’s Matrix-Tree Theorem (MTT) [Tutte, 1984; Koo et al., 2007; Smith and Smith, 2007] to induce the latent structure of an input sentence. Such latent structure can be viewed as multiple full dependency forests, which efficiently represent all possible dependency trees within a compact and dense structure.

Given a sentence \( s = w_1, ..., w_n \), where \( w_i \) represents the \( i \)-th word, we define a graph \( G \) on \( n \) nodes, where each node refers to the word in the \( s \), and the edge \((i, j)\) refers to the dependency between the \( i \)-th word (head) to the \( j \)-th node (modifier). We denote the contextual output of the sentence \( h \in \mathbb{R}^{n \times d} \) as \( h = h_1, ..., h_n \), where \( h_i \in \mathbb{R}^d \) represents the hidden state of the \( i \)-th word with a \( d \) dimension. We use the bidirectional LSTM to obtain contextual representations of the sentence.

For the graph \( G \), MTT takes the edge scores and root scores as inputs then generates a latent forest by computing the marginal probabilities for each edge. Given the input \( h \) and a weight vector \( \theta^k \in \mathbb{R}^m \) of dependencies, where \( m \in \mathbb{R} \) represents the number of dependencies for the \( k \)-th \( (k \in [1, N]) \) latent structure, inducing the \( k \)-th latent forest for the input \( h \) amounts to finding the latent variables \( z_{ij}^k(h; \theta^k) \) for all edges that satisfy \( i \neq j \) and root node whose index equals to 0.

The \( k \)-th latent forest induced by MTT contains many non-projective dependency trees, which are denoted by \( T^k \). Let \( P(y|h; \theta^k) \) denote the conditional probability of a tree \( y \) over \( T^k \). Following the formulation by Koo et al. [2007], the marginal probability of a dependency edge from \( i \)-th word to \( j \)-th word for the \( k \)-th forest can be expressed as:

\[
P(z_{ij}^k = 1) = \sum_{y \in T^k:(i,j) \in y} P(y|h; \theta^k) \tag{1}
\]

We derive two steps to obtain the marginal probabilities expressed in Equation (1).

### Computing Attention Scores

Inspired by Vaswani et al. [2017], we calculate the edge scores by the multi-head attention mechanism, which allows the model to jointly attend to information from different representation subspaces. \( N \) attention matrices will be fed into the MTT to obtain \( N \) latent forests in order to capture different dependencies in different representation subspaces. The attention matrix for the \( k \)-th head is calculated by a function of the query \( Q \) with the corresponding key \( K \). Here \( Q \) and \( K \) are both equal to the contextual representation \( h \). We project \( Q \) and \( K \) to different representation subspaces in order to generate \( N \) attention matrices for calculating \( N \) latent forests. Formally, the \( k \)-th forest \( S^k \) is given by:

\[
S^k = \text{softmax}(\frac{QW^Q \times (KW^K)^T}{\sqrt{d}}) \tag{2}
\]
where $\mathbf{W}^Q \in \mathbb{R}^{d \times d}$ and $\mathbf{W}^K \in \mathbb{R}^{d \times d}$ are parameters for projections. $\mathbf{S}^k_i$ denotes the normalized attention score between the $i$-th token and the $j$-th token with $\mathbf{h}_i$ and $\mathbf{h}_j$. Then, we compute the root score $r^k_i$, which represents the normalized probability of the $i$-th node to be selected as the root node of the $k$-th forest:
\[
    r^k_i = \mathbf{W}^k_i \mathbf{h}_i
\]
where $\mathbf{W}^k_i \in \mathbb{R}^{d \times d}$ is the weight for the projection.

### Imposing Structural Constraint
Following Koo et al. [2007] and Smith and Smith [2007], we calculate the marginal probability of each dependency edge of the $k$-th latent forest, by injecting a structural bias on $\mathbf{S}^k_i$.

We assign non-negative weights $\mathbf{P}^k \in \mathbb{R}^{n \times n}$ to the edges as:
\[
P^k_{ij} = \begin{cases} 
0 & \text{if } i = j \\
\exp(S^k_{ij}) & \text{otherwise}
\end{cases}
\]
where $\mathbf{P}^k_{ij}$ is the weight of the edge between the $i$-th and the $j$-th node. We define a Laplacian matrix $\mathbf{L}^k \in \mathbb{R}^{n \times n}$ of $\mathbf{G}$ in Equation (5), and its variant $\hat{\mathbf{L}}^k \in \mathbb{R}^{n \times n}$ in Equation (6).

\[
\mathbf{L}^k_{ij} = \begin{cases} 
\sum_{v=1}^{n} \mathbf{P}^k_{i,j} & \text{if } i = j \\
-\mathbf{P}^k_{i,j} & \text{otherwise}
\end{cases}
\]

\[
\hat{\mathbf{L}}^k_{ij} = \begin{cases} 
\exp(r^k_i) & \text{if } i = 1 \\
\mathbf{L}^k_{ij} & \text{if } i > 1
\end{cases}
\]

We use $\mathbf{A}^k$ to denote the marginal probability of the dependency edge between the $i$-th node and the $j$-th node. Then, $\mathbf{A}^k_{ij}$ can be derived based on:
\[
\mathbf{A}^k_{ij}(z_{ij} = 1) = (1 - \delta_{i,j})\mathbf{P}^k_{ij}[(\hat{\mathbf{L}}^k)^{-1}]_{ij} - (1 - \delta_{i,j})\mathbf{P}^k_{ij}[(\hat{\mathbf{L}}^k)^{-1}]_{ji}
\]
where $\delta$ is Kronecker delta and $\mathbf{A}^k \in \mathbb{R}^{n \times n}$ can be interpreted as a weighted adjacency matrix for the $k$-th forest. Now we can feed $\mathbf{A} \in \mathbb{R}^{n \times n}$ into the forest encoder to update the representations of nodes in the latent structure.

### Adaptive Pruning Strategy
Prior dependency-based model [Zhang et al., 2018] also proposes rule-based method to prune a dependency tree to further improve relation classification performance. However, the weighted adjacency matrix $\mathbf{A}$ is derived based on a continuous relaxation, and such induced structures are not discrete, so the existing rule-based pruning methods are not applicable. Instead, we use $\alpha$-entmax [Blondel et al., 2018; Correia et al., 2019] to remove irrelevant information by imposing the sparsity constraints on the adjacency matrix. $\alpha$-entmax is able to assign exactly zero weights. Therefore, an unnecessary path in the induced latent forests will not be considered by the latent forest encoder. The expression of our soft pruning strategy is described as:
\[
\mathbf{A}^k = \alpha\text{-entmax}(\mathbf{A}^k)
\]
where $\alpha$ is a parameter to control the sparsity of each adjacency matrix. When $\alpha=2$, the entmax recovers the sparsemax mapping [Martins and Astudillo, 2016]. When $\alpha=1$, it recovers the softmax mapping. Correia et al. [2019] propose an exact algorithm to learn $\alpha$ automatically. Here we apply $k$-$\alpha$-entmax to $k$ latent forests, which enables the model to develop different pruning strategies for different latent forests.

### 2.2 Forest Encoder
Given $N$ latent forests generated by the forest inducer, we encode them by using densely-connected graph convolutional networks [Kipf and Welling, 2017; Guo et al., 2019b]. Formally, given the $k$-th latent forest, which is represented by the adjacency matrix $\mathbf{A}^k$, the convolution computation for the $i$-th node at the $l$-th layer, which takes the representation $\mathbf{h}_i^{l-1}$ from previous layer as input and outputs the updated representations $\mathbf{h}_i^l$, can be defined as:
\[
\mathbf{h}_i^l = \sigma(\sum_{j=1}^{n} \mathbf{A}^k_{ij} \mathbf{W}_k^l \mathbf{h}_j^{l-1} + \mathbf{b}_k^l)
\]
where $\mathbf{W}^l_k$ and $\mathbf{b}^l_k$ are the weight matrix and bias vector for the $k$-th latent forest in the $l$-th layer, respectively. $\sigma$ is an activation function. $\mathbf{h}_0^l \in \mathbb{R}^{d}$ is the initial contextual representation of the $i$-th node. Then a linear combination layer is used to integrate representations of the $N$ latent forests:
\[
\mathbf{h}_\text{comb} = \mathbf{W}_\text{comb} \mathbf{h}_\text{out} + \mathbf{b}_\text{comb}
\]
where $\mathbf{h}_\text{out}$ is the output by concatenating outputs from $N$ separated convolutional layers, i.e., $\mathbf{h}_\text{out} = [\mathbf{h}^{(1)}; \ldots; \mathbf{h}^{(N)}] \in \mathbb{R}^{d \times N}$. $\mathbf{W}_\text{comb} \in \mathbb{R}^{(d \times N) \times d}$ is a weight matrix and $\mathbf{b}_\text{comb}$ is a bias vector for the linear transformation.

### 3 Experiments

#### 3.1 Data
We evaluate our LF-GCN model with four datasets on two tasks, namely cross-sentence $n$-ary relation extraction and sentence-level relation extraction.

For cross-sentence $n$-ary relation extraction, we use two datasets generated by Peng et al. [2017], which has 6,987 ternary relation instances and 6,087 binary relation instances extracted from PubMed. The relation label contains five categories, e.g., “sensitivity”, “resistance” and “none”. Following Song et al. [2018], we define two sub-tasks for a more detailed evaluation, i.e., binary-class $n$-ary relation extraction and multi-class $n$-ary relation extraction. For binary-class extraction, we cluster the four relation classes as “Yes” and treat the label “None” as “No”.

For sentence-level relation extraction, we follow the experimental settings by Lifeng et al. [2020] on BioCreative Vi CPR [Krallinger et al., 2017] and Phenotype-Gene relation (PGR) [Sousa et al., 2019]. The CPR dataset contains the relations between chemical components and human proteins. It has 16,107 training, 10,030 development and 14,269 testing instances, with five regular relations, such as “CPR:3”, “CPR:9” and “None” relation. PGR introduces the relations between human phenotypes with human genes, and it contains 11,780 training instances and 219 test instances, with binary class “Yes” and “No” on relation labels.
We also use SemEval-2010 Task 8 [Hendrickx et al., 2009] dataset from the news domain to evaluate the generalization capability of our model. It has 10,717 instances with 9 types of relations and a special “Other” relation.

### 3.2 Settings

We tune the hyper-parameters according to the results on the development sets. For the cross-sentence n-ary relation extraction task, we use the same data splits as Song et al. [2018], stochastic gradient descent optimizer with a 0.9 decay rate, and 300-dimensional GloVe. The hidden size of both BiLSTM and GCNs are set as 300.

For cross-sentence task, we report the test accuracy averaged over five cross validation folds [Song et al., 2018] for the cross-sentence n-ary task. For the sentence-level task, we report the F1 scores [Lifeng et al., 2020].

### 3.3 Results on Cross-Sentence n-ary Task

To verify the effectiveness of the model in predicting intersentential relations, we compare LF-GCN against state-of-the-art systems on the cross-sentence n-ary relation extraction task [Peng et al., 2017], as shown in Table 1. Previous systems using the same syntax type are grouped together.

**Full Tree:** models use the 1-best dependency graph constructed by connecting roots of dependency trees corresponding to the input sentences. DAG LSTM encodes the graph by using graph-structure LSTM, while GRN and GCN encode it using graph recurrent networks and graph convolutional networks, respectively.

**Pruned Tree:** model with pruned trees as inputs, whose dependency nodes and edges are removed based on rules [Zhang et al., 2018]. GCN is used to encode the resulted structure.

**Forest:** model constructs multiple fully-connected weighted graphs based on the multi-head attention [Vaswani et al., 2017], where the graph can be viewed as a dependency forest.

Models with pruned trees as inputs tend to achieve higher results than models with full trees. Intuitively, longer sentences in the cross-sentence task correspond to more complex dependency structures. Using an out-of-domain parser may introduce more noise to the model. Removing the irrelevant nodes and edges of the parse tree enables the model to perform better prediction. However, a rule-based pruning strategy [Zhang et al., 2018] may not yield optimal performance. In contrast, LF-GCN induces the dependency structure automatically, which can be viewed as a soft pruning strategy learned from the data. Compared to GCN models, our model obtains 2.2 and 2.6 points improvement over the best performing model with pruned trees for the ternary relation extraction. For binary relation extraction, our model achieves accuracy of 86.7 and 87.1 under Single and Cross settings, respectively, which surpasses the state-of-the-art AGGCN model. We believe that our LF-GCN is able to distill relevant information and filter out noises from the representation for better prediction.

### 3.4 Results on Sentence-Level Task

To examine LF-GCN on sentence-level task, we compare LF-GCN with state-of-the-art models on two medical datasets, i.e., CPR [Krallinger et al., 2017] and PGR [Sousa et al., 2019]. These systems are grouped in three types based on the syntactic structure used as shown in Table 2 and Table 3. Results labeled with “*” are obtained based on the retrained models using their released implementations, as we don’t have published results for the dataset.

**None:** models do not use any pre-trained parsers. Random-DDCNN uses a randomly initialized parser [Dozat and Manning, 2017] fine-tuned by the relation prediction loss. Att-GRU stacks a self-attention layer on top of the gated recurrent units and Bran relies on a bi-affine self-attention model to capture the interactions in the sentence. BioBERT is a pre-trained language representation model for biomedical text.

**Tree:** models use the 1-best dependency tree. Full trees are encoded by GCN, GRN and DDCNN, respectively. BOLSTM only encodes words on the shortest dependency path.
KGRN generates a forest by merging with scores greater than a pre-defined threshold. KBest-GRN constructs a dependency forest by keeping all the edges without recourse to any pre-trained parsers, while it outperforms. On the other hand, LF-GCN also makes predictions without any pre-trained parsers, which confirms our hypothesis that the error propagation, which is caused by the low parsing accuracy of an out-of-domain parser, can be alleviated by constructing weighted graphs (forests). Compared with models which encode fixed dependency forests as inputs, our LF-GCN model achieves 58.9 and 91.9 scores on CPR and PGR datasets, which are consistently better than all forest generation approaches. These results suggest that the induced latent structure is able to capture task-specific information for better relation extraction.

### 3.5 Results on News Domain

LF-GCN can also be used in other domains. Table 4 gives the results on SemEval [Hendrickx et al., 2009] dataset from the news domain. Using limited training data, LF-GCN outperforms the models with a dependency tree including Tree-GRN and ForestFT-DDCNN by almost 1 point and is comparable with the models with dependency forests including AGGCN and ForestFT-DDCNN. This demonstrates that LF-GCN is able to learn a comparable expressive structure compared with the structure generated by an in-domain parser. LF-GCN is 0.6 point worse than Edgewise-GRN. The reason is that the parsing performance for newswire is much more accurate than the biomedical domain. We believe that our model is able to achieve higher performance if more training data is available.

### 3.6 Analysis and Discussion

#### Performance against Sentence Length

To investigate our LF-GCN performance under different sentence lengths, we split the test set of CPR into three categories \((0, 25], [25, 50], > 50\) based on the lengths. As shown in Figure 3, we compare our LF-GCN with Tree-GRN [Song et al., 2019] and ForestFT-DDCNN [Lifeng et al., 2020]. In general, LF-GCN outperforms Tree-GRN and ForestFT-DDCNN for each group of instances, showing the effectiveness of our model based on a latent structure induction. The performance gap is enlarged when the instance length increases. Intuitively, longer instances are more challenging since the dependency structure is a more sophisticated tree. These results illustrate that the induced structure is able to capture complex non-local interactions for better prediction.

#### Performance against Number of Forests

Figure 4 shows the performance of LF-GCN and AGGCN with different number of forests, since AGGCN also leverages the adjacency matrices of forests. AGGCN computes attention matrices and treats them as the inputs of the models. Figure 4 demonstrates that LF-GCN is able to capture complex non-local interactions for better prediction.
ages the multi-head attention mechanism [Vaswani et al., 2017] to generate multiple weighted graphs. Even though AGGCN is initialized with the 1-best dependency tree generated by a pre-trained parser, LF-GCN consistently outperforms it under the same number of forest without relying on any parsers, where the numbers range from 1 to 4. These results demonstrate that our model is able to construct informative structures only based on the input text.

3.7 Case Study

In this section, we use the Chu-Liu-Edmonds [Chu and Liu, 1965] algorithm to extract $N$ non-projective dependency trees from $N$ latent forests, where each forest is expressed by a weighted adjacency matrix in Equation 8. Here $N$ equals to 2. We select an instance from the CPR development set, whose relations can be correctly predicted by our LF-GCN.

Case I: As shown in Figure 5, these two dependency trees are able to capture rich interactions between the entities Schisandrin B (index 0 and 1) and DT-diaphorase (index 10, 11 and 12), which have a “up regulator” relation, denoted as “CPR:3”. For example, the token “enhancing” (index 9), which shares the similar semantic as the gold relation “up regulator”, is selected in the path between these two entities. Furthermore, these two trees show different dependencies between tokens, which confirms our hypothesis that inducing multiple forests can include more useful information.

Case II: However, as shown in Figure 6, many dependency trees induced by structure attention are shallow and do not resemble to a linguistic syntax structure. Figure 6 shows two shallow trees extracted from the latent forests before imposing sparsity constraints. We observe that the constructed dependency trees tend to pick the first token of the sentence as the root, and all other tokens as the children. Interestingly, even though such trees have little to no structure, the model is still able to predict the correct relation label. We also notice that adding the $\alpha$-entmax helps to induce deeper and more informative structures. We leave the investigation of this phenomenon as future work.

4 Related Work

Latent Structure Induction: Latent structure models are powerful tools for modeling compositional data and building NLP pipelines [Yogatama et al., 2016; Niculae et al., 2018]. A challenge with structured latent models is that they involve computing an “argmax” (i.e., finding a best scoring discrete structure such as a parse tree) in the middle of a computation graph. Since this operation has null gradients, back propagation cannot be used. There are three main strategies to solve this issue including reinforcement learning, surrogate gradients and continuous relaxation. In this paper, we mainly focus on continuous relaxations, for which the exact gradient can be computed and back propagated [Kim et al., 2017; Liu and Lapata, 2018; Nan et al., 2020].

Medical Relation Extraction: Early efforts focus on predicting relations between entities by modeling interactions in the 1-best dependency tree [Peng et al., 2017; Song et al., 2018]. Recently, dependency forests were used to alleviate the error cascading caused by an out-of-domain parser. Song et al. [2019] build a forest by adding edges and labels that a pre-trained parser is confident about. Lifeng et al. [2020] construct full forests represented by a 3-dimensional tensor generated by a pre-trained parser fine-tuned by the relation prediction loss. Instead of using an out-of-domain parser, our model dynamically induces multiple dependency forests solely based on the medical dataset in an end-to-end manner.

5 Conclusion

In this paper, we propose a novel model that is able to automatically induce a latent structure for better relation extraction, without recourse to any tree supervisions or pre-training. Extensive results on four medical datasets show that our approach is able to better alleviate the error propagation caused by an out-of-domain dependency parser, giving significantly better results than previous state-of-the-art systems.

Acknowledgments

We would like to thank the anonymous reviewers for their thoughtful and constructive comments. This research is supported by Ministry of Education, Singapore, under its Academic Research Fund (AcRF) Tier 2 Programme (MOE AcRF Tier 2 Award No: MOE2017-T2-1-156). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the Ministry of Education, Singapore.
References


