Explore Unsupervised Structures in PLMs for Relation Extraction

Xi Yang*, **Tao Ji***, **Yuanbin Wu**
School of Computer Science and Technology
East China Normal University
{xyang41,taoji}@stu.ecnu.edu.cn, ybwu@cs.ecnu.edu.cn

Abstract

Syntactic trees have been widely applied in relation extraction (RE). However, since parsing qualities are not stable on different text domains and a pre-defined grammar may not well fit the target relation schema, the introduction of syntactic structures sometimes fails to improve RE performances consistently. In this work, we study RE models with various unsupervised structures mined from pre-trained language models (e.g., BERT). We show that, similar to syntactic trees, unsupervised structures are quite informative for RE task: they are able to obtain competitive (even the best) performance scores on benchmark RE datasets (ACE05, WebNLG, SciERC). We also conduct detailed analyses on their abilities of adapting new RE domains and influence of noise links in those structures. The results suggest that unsupervised structures are reasonable alternatives of commonly used syntactic structures in relation extraction models [1].

1 Introduction

Relation extraction (RE) tasks aim to extract pairs of text spans (*entities*), each of which expresses certain *relation* semantics. As shown in Figure 1, w_1w_2 and w_5 are two extracted entities with the type of person (PER) and geography (GPE) respectively, and they carry a physical location relation (PHYS).

While end-to-end neural relation extractors have achieved great success (Wei et al., 2020; Wang et al., 2021b; Ye et al., 2022), proper prior knowledge or external features are still crucial to make RE models more flexible and robust. Syntactic trees (especially, dependency trees) are such features which are widely applied in RE systems (Xu et al., 2015; Miwa and Bansal, 2016; Zhang et al., 2018; Guo et al., 2019; Fu et al., 2019; Mandya et al., 2020; Tian et al., 2021). The prior dependency relations among words are believed to be helpful for spotting most user-defined relations. However, to fulfil their advantages, there are still some challenges to tackle. For example (Figure 1), a syntactic tree usually characterizes the full syntactic structure of a sentence, but some parts of the dense structure may be redundant (even misleading) when detecting sparse relations among entities. Another problem is that since syntactic parsers are trained on external Treebanks, parsing qualities are no longer guaranteed when the Treebanks' text domain mismatch the target RE text domain.

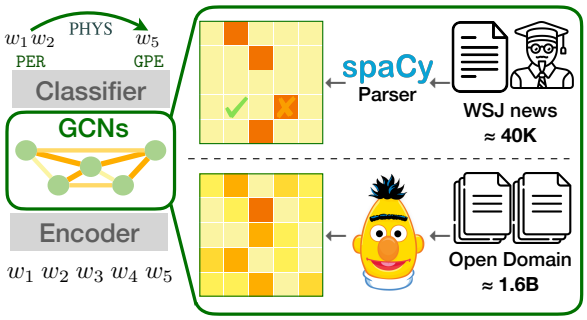

Figure 1: A diagram of a RE model using different external structures. The top is the traditional dependency tree structure and the bottom is unsupervised but informative structure inducted by PLM.

Here, we investigate another kind of external structural features, namely unsupervised structures mined from pre-trained language models (PLMs). Like other NLP tasks, PLMs have been deployed in RE models due to their strong abilities on unsupervisely aggregating semantic information from large corpora and contextual information from the whole input sentence. Current RE models use PLMs to encode words: the classifier queries word embeddings from PLMs as features. We ask that, apart from embedding features, whether RE models could also query PLMs for structural features. On the one side, we would like those structures to be easy to access

*Equal contribution.
[1]Our code is publicly available at https://github.com/xyang41/structures-for-RE.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 6103–6117
December 7-11, 2022

as syntactic structures, on the other side, we would expect them to be able to provide rich syntactic and semantic information as PLM embeddings.

In this paper, we obtain unsupervised structures using three types of interpretation tools (*attention-based*, *occlusion-based* and *gradient-based*). They basically use different kinds of information to characterize connections among input words, and output weighted complete graphs with words as nodes and connection scores as edge weights. Then, we adopt graph convolutional networks (GCNs) as a plug-and-play encoding module to incorporate structures into relation extraction models. We systematically test RE models with unsupervised structures and find that,

- on benchmark datasets (ACE05, WebNLG, SciERC), they indeed provide additional performance gains comparing with vanilla RE feature sets. Furthermore, under a fair setting, they are able to obtain competitive (or better) performances than dependency trees.

- on the scientific RE task (SciERC), we observe that unsupervised structures inherit scalability from PLMs: if the target RE domain is different from the training domain of PLMs, we could fine tune PLMs with unsupervised text of the target RE domain and the structures extracted on the fine-tuned PLMs could fit the RE task better.

Therefore, considering the cost of getting structural features (i.e., with or without human annotations), we think that unsupervised structures are promising features for future RE models.

2 Preliminary

Pre-trained Language Models are important feature extractors in modern NLP systems. In this work, we will take the BERT (Devlin et al., 2019) family as our demonstrative PLMs. Main components of BERT are Transformer blocks (Vaswani et al., 2017) which consist of multiple self-attention heads. For a token x_i in an input sentence X of n tokens $\{x_1, x_2, \ldots, x_n\}$, an attention head computes the token's attention scores over all input tokens and outputs a hidden representation $\boldsymbol{h}_i$. We denote the encoding procedure as $\boldsymbol{h}_i = \texttt{PLM}_i(X)$.

Graph Convolutional Networks (GCN) are powerful feature extractors for structured data. (Kipf and Welling, 2017) . They learn hidden representations for graph nodes by gradually collecting information from their neighbors. In relation extraction, GCNs are popular tools for encoding external syntactic features (Zhang et al., 2018; Fu et al., 2019; Mandya et al., 2020; Tian et al., 2021). We follow Marcheggiani and Titov (2017) by using a bi-directional version of GCN. Denote $\{\boldsymbol{h}_1^{(u)}, ..., \boldsymbol{h}_n^{(u)}\}$ to be intermediate hidden representations of u-th layer of a GCN,

$$\vec{\boldsymbol{h}}_i^{(u+1)} = \texttt{ReLU}\Big(\sum_{j \in \vec{N}(i)} (\vec{\boldsymbol{W}}^{(u)} \boldsymbol{h}_j^{(u)} + \vec{\boldsymbol{b}}^{(u)})\Big)$$

$$\overleftarrow{\boldsymbol{h}}_i^{(u+1)} = \texttt{ReLU}\Big(\sum_{j \in \overleftarrow{N}(i)} (\overleftarrow{\boldsymbol{W}}^{(u)} \boldsymbol{h}_j^{(u)} + \overleftarrow{\boldsymbol{b}}^{(u)})\Big)$$

$$\boldsymbol{h}_i^{(u+1)} = \vec{\boldsymbol{h}}_i^{(u+1)} \oplus \overleftarrow{\boldsymbol{h}}_i^{(u+1)},$$

where $\boldsymbol{W}, \boldsymbol{b}$ are parameters of GCN layers, $N(i)$ contain neighbors of node i, $\{\rightarrow, \leftarrow\}$ represent the outgoing and incoming direction of a node respectively, and $\oplus$ is the vector concatenation operator.

3 How to Get Structures

We describe the structures included in our RE models. One kind of structure is derived from dependency trees, and another kind is mined from PLMs. Both of them are represented by a $n \times n$ importance matrix G (also called saliency map). An entry G_{ij} of G indicates the strength of connection between word x_j and word x_i.

3.1 Dependency Structures

Dep We obtain dependency trees of sentences with an off-the-shelf dependency parser (spaCy Honnibal and Johnson, 2015). The matrix G is the adjacency matrix of the tree,

$$G_{ij} = \begin{cases} 1 & \text{if edge } x_i \rightarrow x_j \text{ exists} \\ 0 & \text{otherwise} \end{cases}. \quad (1)$$

GP-Dep It has been shown that with a proper pruning of dependency trees, RE models may get a more clear view of task-specific structures. We apply the pruning strategy proposed in Zhang et al. (2018). The dependency tree is first pruned into a subtree rooted at the lowest common ancestor (LCA) of all gold entities, and then only paths from LCA to each entity and edges in the subtree which are directly connected to these paths are retained. It is worth noting that the pruning strategy assumes gold entities are given, which is a *cheating* strategy when we perform joint entity relation extraction.

3.2 Unsupervised Structures in PLM

To acquire unsupervised structures from PLMs, we explore three popular model interpretation methods,

- Attention-based structures (Abnar and Zuidema, 2020),
- Occlusion-based structures (Wu et al., 2020),
- Gradient-based structures (Li et al., 2016; Denil et al., 2014; Sundararajan et al., 2017).

3.2.1 Attention-based Structures

The attention mechanism is the core component of PLMs, as it allows each input word to interact with all words in the input sequence and find out which they should pay more attention to. Previous work shows that attention in PLMs could characterise some of the syntax rules, and the span boundary of entities (Clark et al., 2019). These could be useful for the RE task.

Attention matrix (Attn) At l-th layer, attention matrix A^l describes the attention score between any two words. [2] The A^l could be treated directly as an importance matrix G:

$$G = A^l. \tag{2}$$

Rollout and Flow Abnar and Zuidema (2020) show that word representation aggregating from other words gets increasingly mixed across multiple layers, which makes attention scores unreliable. Thus, they propose two methods for approximating the attention to individual input words, namely *rollout* and *flow*. **Rollout** tracks the propagation of information from the input layer to the hidden vectors in the higher layers. The l-th layer's rollout matrix $\tilde{A}^l$ is obtained by recursively multiplying the attention matrices $A_{\leq l}$ in all the layers below.

$$G = \tilde{A}^l = \begin{cases} A^l \cdot \tilde{A}^{l-1} & \text{if } l > 1 \\ A^1 & \text{if } l = 1 \end{cases} \tag{3}$$

Flow runs the max-flow $\texttt{Flow}^l(V, E, src, tgt)$ algorithm to obtain the importance matrix, where the set of nodes is the words in the sentence ($V = X$), the set of edges is the union of the attention matrices from 1-th layer to l-th layer ($E = A^1 \cup \cdots \cup A^l$), the source node is x_i and the target/sink node is x_j.

$$G_{ij} = \texttt{Flow}^l(X, A^1 \cup \cdots \cup A^l, x_i, x_j) \tag{4}$$

[2] By convention, we obtain the single matrix by averaging multi-head attentions.

3.2.2 Occlusion-based Structures

Occlusion-based methods measure the impact of masking one word on the representation of another word. Previous studies observe strong inter-chunk word impacts in noun phrases and verb phrases (Wu et al., 2020), which may improve the RE model.

Perturbed Masking (PMask) Wu et al. (2020) apply two perturbations to an input sentence and then compare the differences in the PLMs outputs. They use $X_{-\{x_i\}}$ to denote the word x_i is replaced with the [MASK] token, and $X_{-\{x_i,x_j\}}$ to denote the words x_i, x_j both replaced by the [MASK] token. The influence of word x_j on word x_i can be measured by the change in the hidden vector of the two perturbations.

$$G_{ij} = d\left(\texttt{PLM}_i(X_{-\{x_i\}}), \texttt{PLM}_i(X_{-\{x_i,x_j\}})\right) \tag{5}$$

Where $d(\cdot, \cdot)$ is the euclidean distance metric.

3.2.3 Gradient-based Structures

The gradient-based method is a standard interpretation approach in neural networks (Dimopoulos et al., 1995; Li et al., 2016), by calculating partial derivatives of the prediction function. The scale of the derivative is correlated with the influence of the derivative term to the prediction. When applying it to PLMs, the prediction is MLM objective and the derivative term is each word. Theoretically it could identify the words which are important for predicting entities, and the words which are very dependent on entities when predicting themselves.

Gradient (Grad) uses the gradient of the MLM objective function $\mathcal{L}_{\texttt{MLM}}(X_{\{-x_i\}})$ on masked token x_i for token x_j as the importance score G_{ij}. With this definition, the G_{ij} is simply the squared L2-norm of the prediction function's gradient w.r.t. the word embedding, i.e.,

$$G_{ij} = \left\| \nabla_{x_j} \mathcal{L}^i_{\texttt{MLM}}(X_{\{-x_i\}}) \right\|_2^2 \tag{6}$$

Gradient×Input (GInput) A slight variation of gradient approach uses partial derivatives multiplied by the variable's value (Denil et al., 2014), considering the effect of different input tokens. Hence, the importance score G_{ij} is a dot product of the gradient and the word embedding:

$$G_{ij} = \left(\nabla_{x_j} \mathcal{L}^i_{\texttt{MLM}}(X_{\{-x_i\}}) \right)^\top \cdot \boldsymbol{x_j} \tag{7}$$

Methods	Complexity	Num	Speed
SpaCy → Dep	$\mathcal{O}(n)$	1	207
UDPipe → Dep	$\mathcal{O}(n^3)$	1	46
Attn	F	L	**1675**
Rollout	$2F$	L	**1436**
Flow	nLF	L	**838**
PMask	n^2F	1	25
Grad	$n(F+B)$	1	83
GInput	$n(F+B)$	1	79
GInteg	$kn(F+B)$	1	14

Table 1: Time complexity and number of candidate structures for different structure mining methods. F or B denotes the complexity of one forward or backward propagation, n is the sentence length, L is the number of layers in a PLM, and k denotes the number of interpolations in the **GInteg**. **Bold** indicates speed over the dependency tree.

Integrated Gradients (GInteg) A more recent variation of gradient approach is called Integrated Gradients (Sundararajan et al., 2017). It computes input attribution by aggregating gradients along a linear path between the input feature and the baseline feature (usually a zero vector). It offers two desirable theoretical guarantees motivating its usage: sensitivity and implementation invariance [3]. Here we use the [MASK] vector as the baseline vector $\boldsymbol{b}$. To calculate the importance score G_{ij}, we apply a linear interpolation between $\boldsymbol{b}$ and x_j to obtain the integration gradient.

$$G_{ij} = (\boldsymbol{x_j} - \boldsymbol{b}) \times \int_{\alpha=0}^{1} \frac{\partial \mathcal{L}^i_{\texttt{MLM}}\left(X_{\{-x_i, -\alpha \times x_j\}}\right)}{\partial \boldsymbol{x_j}} d\alpha \tag{8}$$

where $\alpha = 0$ means that x_j is represented by the original word vector and $\alpha = 1$ means that x_j is masked. In the implementation, we calculate Eq 8 by discrete approximation (choose $k = 10$ interpolations evenly).

3.2.4 Comparison

We compare the different structure mining methods on time complexity and the number of candidate structures (Table 1). For time complexity, the attention-based methods can obtain the attention matrix in one forward propagation, and notably, the **Flow** suffers from the maximum flow algorithm. The occlusion-based method has to do n^2 times MLM. The gradient-based methods have to do n times MLM as well as n times back-propagation, and in particular, the **GInteg** does k extra interpolations. Here we provide two classical dependency parsers as references, the transition-based parser SpaCy, with a linear decoding algorithm, and the graph-based parser UDPipe, with a maximum span tree decoding algorithm. Overall, in terms of time costs, **Attn**<**Rollout**<**Flow**<**SpaCy**<**Grad**≈**GInput**<**UDPipe**<**PMask**<**GInteg**. Moreover, attention-based approaches produce one candidate structure at each layer, while the other approaches produce only one structure.

[3] Refer to Sundararajan et al. (2017) for more details.

Dataset	Train	Dev	Test	#Ent	#Rel
ACE05	10051	2424	2050	7	6
SciERC	1861	275	551	6	7
WebNLG	5019	500	703	-	171

Table 2: The statistics of relation extraction datasets. #Ent and #Rel represent the number of predefined entity and relation types respectively.

4 Relation Extraction Model

RE Model We apply different structures in the setting of *joint entity relation extraction*: given an input, the RE model extracts both entities and relations among the extracted entities. The encoder part (Figure 1) of our model is a stack of BERT and GCN layers (with external structures). After obtaining hidden vector representations of each token, we apply a simple joint extraction procedure from (Sun et al., 2019). Specifically, the model first performs a token-level sequence labelling for extracting entities. Then, for each possible pairs of entities, it builds a representation of the pair by aggregating representations of left tokens, right tokens and in-between tokens regarding that pair. The relation extractor consumes the entity pair representation and performs a multi-class classification.

Datasets We run experiments on three widely used datasets: ACE05[4], SciERC[5] and WebNLG[6]. Regarding text domains, ACE05 collects texts from news, SciERC texts are from AI conference proceedings, and WebNLG are obtained by crowdsourcing with DBPedia relation triples as guidance.

[4] https://github.com/tticoin/LSTM-ER
[5] http://nlp.cs.washington.edu/sciIE/
[6] https://github.com/weizhepei/CasRel

Dataset statistics are shown in the Table 2. More details are described in Appendix A.

Metrics Following previous works, we evaluate RE models with micro Precision, Recall, and F1 score. For ACE05 and SciERC, we judge a correct extraction with *strict evaluation* criteria (Miwa and Bansal, 2016; Sun et al., 2019; Wang et al., 2021b). For WebNLG, we use *partial matching* criteria (Zeng et al., 2018; Fu et al., 2019; Wei et al., 2020). The *partial matching* criteria considers only the head words of the entity pair and ignores the full entity span and types.

Choices of PLMs We choose `bert-base-cased` (Devlin et al., 2019) for ACE05 and WebNLG, and `scibert-basevocab-cased` (Beltagy et al., 2019) for SciERC [7]. The generated structures are fixed and remain unchanged when fine-tuning the RE model.

Hyper-parameters We select hyper-parameters according to F1 score on ACE05 development set with dependency tree structure (**Dep**). We keep the same settings on all datasets and structures. Details of hyper-parameters are listed in Appendix B. We run each experiment 3 times and report averaged F1 scores.

Comparison Settings We design four RE models as control groups to explore the effects of unsupervised structures.

1. *Baseline*, models without structural features and GCN Layers. We note that performances of our baseline RE model is a little higher than the reference model ENPAR (Wang et al., 2021a), and also comparable with other advanced RE models (Wang et al., 2021b; Shen et al., 2021), as shown in Table 3.

2. *Light* structures are self-contained structures given the input sentence: they don't rely on external tools or resources. We consider two types of light structures,

   - **Self-Loop**, each token connects to itself. It is easy to see that GCNs with Self-Loop are equal to *Baseline* with additional parameters stacked for classification heads.
   - **Linear**, each token points to the next token in the sequence.

3. *Dependency-based* structures are described in Section 3.1.

4. *Unsupervised* structures are structures described in Section 3.2.

[7] Most RE works use `scibert-scivocab-uncased` on SciERC dataset for better performance, however, here we adopt `scibert-basevocab-cased` on account of subsequent domain adaptation experiments compared with `bert-base-cased`.

Dataset	Model	Encoder	F1
ACE05	TriMF (Shen et al., 2021)	BERT_B	62.8
	UniRE (Wang et al., 2021b)	BERT_B	64.3
	PL-Marker (Ye et al., 2022)	BERT_B	**66.5**
	ENPAR (Wang et al., 2021a)	BERT_B	62.2
	Baseline	BERT_B	62.5
SciERC	UniRE (Wang et al., 2021b)	SciBERT_S	36.9
	PL-Marker (Ye et al., 2022)	SciBERT_S	**41.6**
	Baseline	SciBERT_B	36.6
WebNLG	HBT (Wei et al., 2020)	BERT_B	91.8
	TPLinker (Wang et al., 2020)	BERT_B	91.9
	PFN (Yan et al., 2021)	BERT_B	**93.6**
	Baseline	BERT_B	91.4

Table 3: Relation extraction performance of *Baseline* compared with other models. Encoders used in different datasets: BERT_B = $\text{BERT}_{\text{Base}}$, SciBERT_S = $\text{SciBERT}_{\text{Scivocab}}$, SciBERT_B = $\text{SciBERT}_{\text{Basevocab}}$. **Bold** in the table indicates the highest relation F1 for each dataset.

5 Analyses and Discussions

5.1 Can Unsupervised Structures Help RE Task? (Unsupervised vs. Baseline)

Our first question is whether unsupervised structures contain useful information for RE models. We compare RE models with and without seven unsupervised structures in Figure 2.

We find that among 21 (7 structures, 3 datasets) RE models with unsupervised structures, 20/21 of them improve relation F1 score over RE models without them. In particular, 15/21 of them outperform the baseline even for the worst values in multiple experiments (see lower bounds of error bars), suggesting that their effectiveness is consistent. Furthermore, one of **Attn** structures achieves the best results on the ACE05 (+1.60 F1) and SciERC (+0.84 F1), [8] and one of **Rollout** structures

[8] We also find that the average results attention-based structures over 12 layers also outperform baseline. See Table 5 for average scores of attention-based structures.

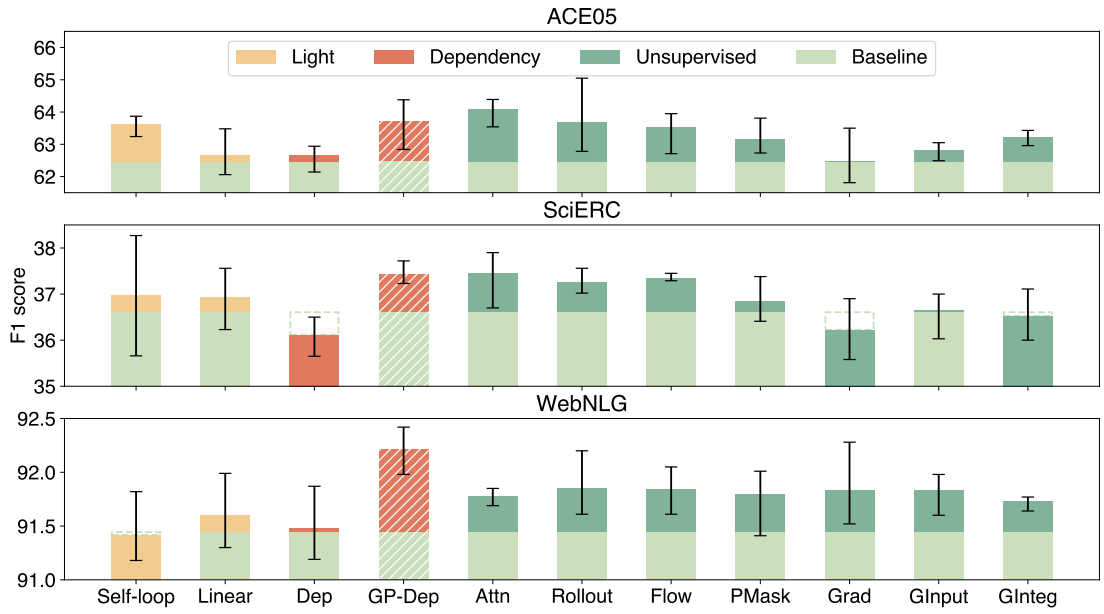

Figure 2: Visualization of relation extraction performance for all structures. Where the darker part above the baseline indicates the improvement, and the ones below represents the specific performance. The bar of GP-Dep structure is filled with slashes (it uses information of gold entities). The two ends of each black vertical line indicates the minimum and maximum performance of three random seeds, and the height of each bar is the mean. See Table 5 in Appendix for exact numbers.

achieves the best results on the WebNLG (+0.41 F1). Overall, the general improvement of unsupervised structures over the baseline suggests their effectiveness in RE tasks.

5.2 Are Dependency Trees the Optimal Structures? (Unsupervised vs. Dep)

After observing unsupervised structures' effectiveness on RE tasks, our next question is whether they are comparable to the widely applied dependency trees. To this end, we compare unsupervised structures with **Dep** structure.

We first find (Figure 2) that regarding the improvement over the baseline (models without structural features), dependency trees are not as effective as unsupervised structures. In fact, comparing with the baseline, **Dep** has a small improvement on ACE05 and WebNLG, and is even 0.49 F1 lower than the baseline on SciERC. We guess the reason of such inefficiency might be redundancy in structures (complete dependency trees may contain unnecessary syntactic relations and make the learning process harder (ACE05, WebNLP)), or poor parsing quality on a different domains (SciERC). We will return to the domain shifting problem in Section 5.6.

On the other side, 20/21 unsupervised structures outperform the complete dependency trees. Specifically, on ACE05, WebNLG and SciERC, the F1 of the best unsupervised structure is 1.42, 1.33 and 0.37 higher than the **Dep** setting, respectively. Surprisingly, on both ACE05 and SciERC, **Attn** is even superior to the "cheating" dependency trees (**GP-Dep**), which are pruned based on gold entity sets. These results indicate that unsupervised structures are promising alternatives of dependency trees in RE models.

5.3 Do Improvements Come from Bigger Models? (Unsupervised vs. Light)

Since models with structural features need additional GCN parameters, we may question whether performance gains come from the new parameters. To make a fair comparison, we introduce *light* structures. They either add new parameters without meaningful structure features (**Self-Loop**), or add shallow word order features (**Linear**).

In Figure 2, we indeed observe performance gains from additional GCN parameters: by comparing six (2 structures, 3 datasets) *light* structure models with baselines, 5/6 of them are better. Moreover, the effect of adding more parameters could also surpass the effect of introducing dependency trees (5/6). Hence, to correctly evaluate structural features, it is important to make models in the same scale.

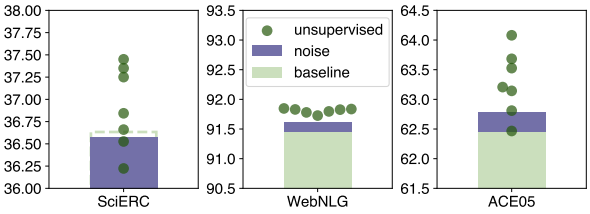

Figure 3: Comparison of the noise-filled structure, the non-noise structure (i.e., baseline) and seven partial-noise (unsupervised) structures on SciERC, WebNLG and ACE05.

Regarding unsupervised features, 16/21 structures are better than at least one *light* structure, and 12/21 of them perform better than all *light* structures. Specifically, on ACE05, **Attn** and **Rollout** perform better than the best *light* structure by 0.46 and 0.06 F1, respectively. On SciERC and WebNLG, almost all unsupervised structures are much better than all of them. Under the fair comparison, we can argue that better relation extraction results brought by applying unsupervised structures are due to these structures themselves, rather than the introduction of extra parameters in GCN modules.

5.4 How About Noise in Structures? (Noise vs. Baseline & Unsupervised)

Learned without explicit structure regularities, unsupervised structures inevitably contain noise information. To better understand those structures (and their roles in RE models), we should understand noise in them better. However, it is hard to characterize noise given no reference of "clean" structures. Here, we design a control group which are random structures (*full-noise*), that is, complete graphs with edge weights randomly generated in $[0, 1]$. Baselines are *zero-noise* models as they don't include any structure. Unsupervised structures are *partial-noise* models as they both contain noise and useful information. We would compare their performances on RE tasks to explore how noise in structures influence performances (Figure 3).

Firstly, we find that, surprisingly, random structures do not obviously hurt the RE model and even outperform zero-noise structure on the two large datasets (WebNLG and ACE05). We guess that it is partly because of the new GCN parameters (like *light* structures), and partly because in large datasets, RE models could also benefit from random noise structure as a regularization (like dropout). Secondly, unsupervised structures generally outperform random structures, where the best structure delivers a significant improvement (+0.87, +0.24, +1.30 on SciERC, WebNLG, ACE05 respectively). Therefore, if we can cancel the effect of noise by comparing against random structures, we would think that information left in unsupervised structures is indeed helpful for RE tasks.

5.5 Which Unsupervised Structure Should We Choose? (Unsupervised vs. Unsupervised)

One practical problem is how to choose unsupervised structure for RE models. Here, we focus on comparing unsupervised structures and analyzing where the differences come from (Figure 2).

On ACE05 and SciERC, **Attn** achieves the highest F1, while the gradient family has limited overall improvements. For WebNLG, all unsupervised structures perform well and do not differ significantly due to the strong RE baseline (91.44 F1). In general, the attention-based structures always bring improvements to RE tasks, while occlusion-based and gradient-based methods do not provide a consistent improvement. Specifically, although **Rollout** and **Flow** are variants of **Attn** with advanced consideration on interpretability, we only observe a slight improvement for the RE task (on WebNLG). While for gradient-based structures, **GInput** and **GInteg** try to improve interpretability over the basic **Grad**, they also improve RE performances against **Grad** on RE tasks.

According to existing researches on model interpretability, which methods are better is still under debate (Serrano and Smith, 2019; Jain and Wallace, 2019; Wiegreffe and Pinter, 2019) , and it remains a challenge to fully investigate differences among unsupervised structures. Here we speculate that unsupervised structures' difference on the RE task is mainly attributed to the faithfulness of the structure generation approach to PLMs (Jacovi and Goldberg, 2020). Attention-based structures are based on attention matrices, and thus are more likely to be faithful to syntactic or semantic structure induced from large-scale unsupervised training. While the other two kinds of structures, occlusion-base and gradient-based, depend on mask language model objectives and require manually designed interventions and approximations (e.g., Taylor expansions in gradient-based methods). The faithfulness of their importance scores assumes that approximations are faithful, which may not be true in RE scenarios.

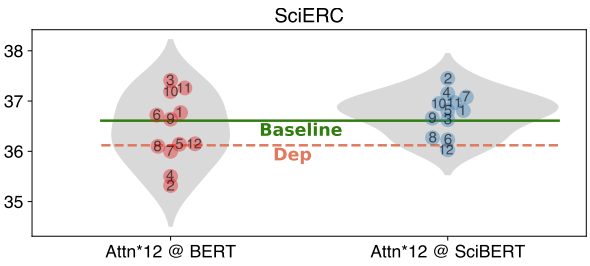

Figure 4: Performance of 12 layers **Attn** structures generated from SciBERT (blue points) compared to that generated from BERT (red points) on the SciERC dataset. The number inside the point represents the ID of the layer. The grey areas depict their distribution characteristics, where SciBERT has a lower variance and a higher mean. The SciBERT we used is obtained by continuous fine-tuning BERT on a large scale of plain texts in the scientific domain.

In summary, for practitioners, we would recommend the attention-based structures for getting a better RE performance scores. If one also takes efficiency into account, the **Attn** structure is the fastest (8 times faster than **Dep**).

5.6 Adapting to New Domains (SciBERT vs. BERT)

It is known that off-the-shelf trained dependency parsers may perform poorly when meeting texts with different domains to Treebanks. In fact, our experiments on SciERC also witness their poor domain adaptation ability (Section 5.2). To adapt dependency parsers to new domains, one may need new dependency tree annotations of new domains, which are expensive to obtain. On the other side, the adaptation of PLMs only require raw text data. Therefore, it is interesting to check whether unsupervised structures inherit this nice property. In other words, whether unsupervised structures mined from adapted PLMs enjoy better domain adaptation performances than those mined from original PLMs.

Here, we use SciERC to study this problem. [9] We compare structures mined from BERT and those mined from SciBERT, which is obtained by fine tuning BERT on additional scientific texts. The results are shown in the Figure 4. We see that 7/12 of runs, structures from SciBERT have better performances than those from BERT. Therefore, an approach for improving adaptability of unsupervised structures is simply adding raw texts in the specific domain to fine tune PLMs, and then the adaptability of structures are automatically enhanced.

5.7 Case Study

To provide more proof of the effectiveness of unsupervised structures, we conduct some case study [10], and show the different focus between the dependency tree and unsupervised structures.

Exampe 1: unsupervised structures ignore strong syntactic relations For the sentence, 'Asked who should govern Iraq, Al-Douri said:" I have nothing to do with that," ' the model with **Dep** predicts a relation: ("who", "Iraq", ORG-AFF), while the unsupervised structure **Attn** predicts nothing. In the dependency tree, ("govern", "who"), ("govern", "Iraq") are two valid arcs which give strong support for the wrong relation ("who", "Iraq", ORG-AFF), while in **Attn**, linking scores (0.007 and 0.008) of the two arcs are weak (as a comparison, linking scores of ("who", ",") is 0.122, ("Iraq", ",") is 0.13), thus the strong syntactic relations which are ignored by the unsupervised structure. It has been shown that PLMs sometimes place more attentions to special tokens (period, comma, SEP) than syntactic heads, and we think that in IE tasks, it also helps filtering suspicious relations.

Example 2: unsupervised structures spot long distance relations For the sentence, 'Beside meeting with Annan, Al-Douri spent several hours during the late morning and afternoon meeting with ambassadors and diplomats in the Delegate's Lounge.' the model with **Dep** predicts nothing, while the unsupervised structure **Attn** predicts a relation: ("Al-Douri", "Delegate's Lounge", PHYS). The two entities in the gold relation are distant both in sentences (17) and in the dependency tree (the path connecting them has a length of 7), while in **Attn**, the linking score (0.112) between them is very high (as a comparison, linking scores of "Al-Dour" with other tokens are less than 0.002). Therefore, unsupervised structures may incorporate new structural features for IE tasks.

6 Related Work

Dependency Trees for RE Many studies (Xu et al., 2015; Miwa and Bansal, 2016; Zhang et al., 2018; Guo et al., 2019; Fu et al., 2019; Mandya

[9] For simplicity, here we only do the comparison on the **Attn** structure of 12 layers in unsupervised structures.

[10] Sentences used in the examples are from ACE05 dataset, and from all unsupervised structures we choose the most representative one **Attn** here.

et al., 2020; Tian et al., 2021) conclude the usefulness of syntactic information for the improvement of RE, e.g., the dependency tree. Because they believe that it can capture long-range dependency, so as to help a model better understand the context. However, dependency trees also contain lots of useless edges for RE task as much noise. To reduce it, variety of pruning methods on dependency trees are proposed.

Xu et al. (2015) only model the information along the shortest dependency path (SDP) between two entities by LSTM. Miwa and Bansal (2016) prune the whole dependency tree into a subtree, which root is the lowest common ancestor (LCA) of two entities. While Zhang et al. (2018) adopt GCN and make a trade-off between them, by reserving edges in LCA subtree which are involved or directly connected to the SDP. Based on that, Mandya et al. (2020) study three subgraphs separately, which consist of SDP and edges directly connected to each entity respectively. The methods mentioned above are all "hard-pruning", which deletes edges completely. Whereas another way considers that much hasty, and applies "soft-pruning" by learning how to prune with additional parameters. Fu et al. (2019) employ another learning stage on relational graph which is the output of studying the whole dependency tree by GCN. Guo et al. (2019) use multi-head attention mechanism (Vaswani et al., 2017) to learn the weight for every edge. And Tian et al. (2021) also introduce the type information of each dependency edge, besides pruning by self-attention.

Besides the noise, the cost of human-annotated labels for dependency trees are quite expensive. Meanwhile, models of generating dependency trees perform unstably on different domains. Therefore, we expect to turn to PLMs for a more easily available structure of some use.

Probe Structures in PLM Representations The success of PLM for many NLG downstream tasks, catches many attentions to the reason of its high performance. Some works probe the representations produced by PLMs, and find that the existing of syntactic knowledge maybe a big reason. For example, Tenney et al. (2019) construct a broad suite of probing tasks to analyse what PLM has learned on huge unsupervised corpus, and discover that there are strong syntactic phenomena in PLM representations. Moreover, Wu et al. (2020) recover syntactic trees from PLM by probing on its representations, and reach a comparable result with dependency trees when applying for fine-grained sentiment classification task.

Based on the above, we propose to extract useful structures in a more efficient way, that is directly from PLM instead of probing the representations, for downstream RE task.

7 Conclusion

We propose to study the performance of RE models with variety of unsupervised structures mined from PLMs. We conducted a series of experiments on RE to verify the effectiveness of them, and analyze their internal compositions and domain adaptation respectively. The main conclusion we reached is unsupervised structures are beneficial to RE, and perform competitive or even better than dependency trees. We also find that unsupervised structures can enhance domain adaptation ability by simply adding raw texts during finetuning PLMs.

Limitations

In this paper, we reach a conclusion that unsupervised structures in PLMs are helpful in relation extraction task. However, there are still some limitations as listed below:

- More theoretical methods are needed to give a accurate explanation for the good performance of unsupervised structures in RE.
- Despite attention-based structures performing best of all unsupervised structures in RE, there are still absent of simple but effective methods on how to choose between different attention layers for using.
- The domain adaptation experiments (Section 5.6) we conducted on unsupervised structures are limited by hardware and time, and still lacking of more systematic studies.
- Although spaCy is a common parser, using different parsers can make our conclusions related to dependency trees more reliable.

Acknowledgement

The authors wish to thank all reviewers for their helpful comments and suggestions and Yufang Liu for her comments on the writing. The corresponding author is Yuanbin Wu. This research was (partially) supported by National Key R&D Program of China (2021YFC3340700) and NSFC(62076097).

References

Samira Abnar and Willem H. Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4190–4197. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@ACL 2019, Florence, Italy, August 1, 2019*, pages 276–286. Association for Computational Linguistics.

Misha Denil, Alban Demiraj, and Nando de Freitas. 2014. Extraction of salient sentences from labelled documents. *CoRR*, abs/1412.6815.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Yannis Dimopoulos, Paul Bourret, and Sovan Lek. 1995. Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Process. Lett.*, 2(6):1–4.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. Graphrel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1409–1418. Association for Computational Linguistics.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 241–251. Association for Computational Linguistics.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1373–1378. The Association for Computational Linguistics.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jiwei Li, Xinlei Chen, Eduard H. Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 681–691. The Association for Computational Linguistics.

Angrosh Mandya, Danushka Bollegala, and Frans Coenen. 2020. Graph convolution over multiple dependency sub-graphs for relation extraction. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6424–6435. International Committee on Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1506–1515. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Yongliang Shen, Xinyin Ma, Yechun Tang, and Weiming Lu. 2021. A trigger-sense memory flow framework for joint entity and relation extraction. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 1704–1715. ACM / IW3C2.

Changzhi Sun, Yeyun Gong, Yuanbin Wu, Ming Gong, Daxin Jiang, Man Lan, Shiliang Sun, and Nan Duan. 2019. Joint type inference on entities and relations via graph convolutional networks. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1361–1370. Association for Computational Linguistics.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yuanhe Tian, Guimin Chen, Yan Song, and Xiang Wan. 2021. Dependency-driven relation extraction with attentive graph convolutional networks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4458–4471. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Yijun Wang, Changzhi Sun, Yuanbin Wu, Hao Zhou, Lei Li, and Junchi Yan. 2021a. ENPAR: enhancing entity and entity pair representations for joint entity relation extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 2877–2887. Association for Computational Linguistics.

Yijun Wang, Changzhi Sun, Yuanbin Wu, Hao Zhou, Lei Li, and Junchi Yan. 2021b. Unire: A unified label space for entity relation extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 220–231. Association for Computational Linguistics.

Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. 2020. Tplinker: Single-stage joint extraction of entities and relations through token pair linking. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 1572–1582. International Committee on Computational Linguistics.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1476–1488. Association for Computational Linguistics.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4166–4176. Association for Computational Linguistics.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. 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, Lisbon, Portugal, September 17-21, 2015*, pages 1785–1794. The Association for Computational Linguistics.

Zhiheng Yan, Chong Zhang, Jinlan Fu, Qi Zhang, and Zhongyu Wei. 2021. A partition filter network for joint entity and relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 185–197. Association for Computational Linguistics.

Deming Ye, Yankai Lin, Peng Li, and Maosong Sun. 2022. Packed levitated marker for entity and relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022, pages 4904–4917. Association for Computational Linguistics.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 506–514. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2205–2215. Association for Computational Linguistics.

A Datasets

Table 2 contains the statistics of three used datasets: ACE05, SciERC and WebNLG. Notably, original WebNLG dataset have 246 predefined relation types without any entity labels, and 171 here is the number of our adopted version ((Wang et al., 2020)). For the generated seven unsupervised structures, we will make them publicly available for download. In addition, due to file size constraints, we only upload the development set on SciERC as supplementary material.

B Hyper-parameters

As mentioned in the main body, we tune all hyper-parameters on ACE05 dataset, then keep the same settings on SciERC and WebNLG. Where the basic network remains consistent with previous work (Sun et al., 2019), we have only tuned the hyper-parameters for the GCN layer. Table 4 shows the value of each hyper-parameter of our relation extraction model for all datasets.

Hyper-parameters	**Default**
Epoch	{100, **200***}
Batch	32
Optimizer	Adam
Learning Rate	2.5e-5
Warm Up Rate	0.2
Scheduler	StepLR
Learning Rate Decay	0.9
Word Embeddings	100
Char Embeddings	50
BERT Dropout	0.01
Hidden Size	768
POS Dimension	15
GCN Layer	{0, 1, **2***, 3}
GCN Dropout	{0, **0.1***, 0.2, 0.3}
Adj Dropout	{ **0***, 0.1, 0.2, 0.3}

Table 4: Our default hyper-parameters settings on ACE05, SciERC and WebNLG, where the results of our hyper-parameter search are in brackets {...} and the * mark the final choice of values.

C Complete Experimental Results

Table 5 and 6 describe our complete experimental results on relation extraction task, including P, R and F1 score for each structure we used in this paper.

As mentioned in the main text, we totally compare the relation extraction performance between four types: *Baseline*, *Light*, *Dependency* and *Unsupervised*. And the left **Random** structure is proposed mainly to simulate the effect of noise in the complete graph. From the Table 5, it is clearly that unsupervised structures reach the best at all evaluations, ignoring the "cheating" structure **GP-Dep**. Moreover, if considering it, *Unsupervised* structures still achieve the best on 2 of 3 datasets. These all convincingly demonstrate the effectiveness of unsupervised structures.

D Results on All PLMs Layers

In this part, we illustrate the entity and relation extraction performance of all layers for attention-based unsupervised structures on ACE05, SciERC and WebNLG datasets, by a scatter chart, as shown in Figure 5. To better depict the performance change of 12 layers for each structure, we use the quadratic curve to fit it.

From those figures, we can clearly see that most of curves are U-shape, i.e., a drop is always occurred in the mid. This indicates that the first and last layers of attention-based structures, are generally outperforming middle layers. Besides, regardless of the shape for each curve, the first layers especially the Layer 1, perform comparable or equal to the best among 12 layers. Therefore, if attention-based structures are expected to apply in the relation extraction task, we recommend the first layer most.

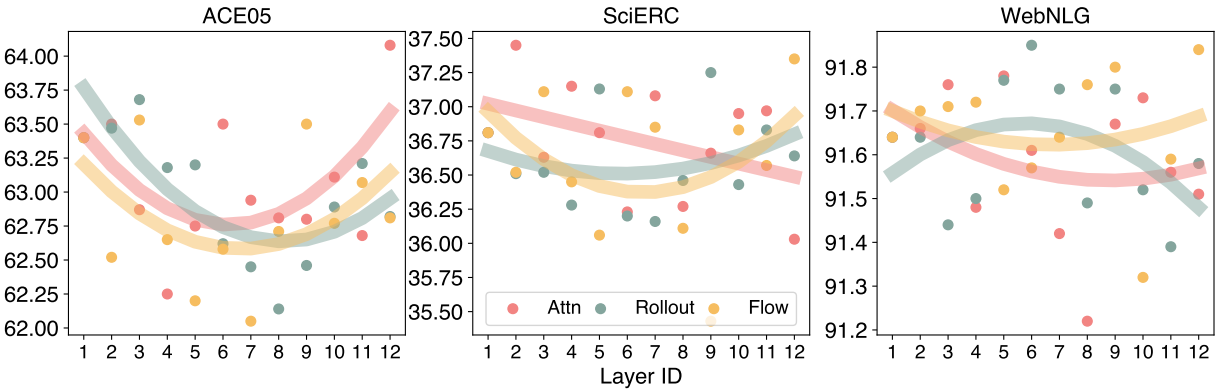

Figure 5: RE performance of attention-based structures for different layers.

Type	Structure	ACE05			SciERC			WebNLG		
		P	R	F1	P	R	F1	P	R	F1
Baseline	-	67.00	58.50	62.46	37.44	35.83	36.61	91.37	91.51	91.44
Light	Self-Loop	67.05	60.53	63.62	37.91	36.12	36.98	91.41	91.42	91.42
	Linear	66.82	58.99	62.66	38.82	35.24	36.94	91.76	91.45	91.60
Dependency	Dep	66.03	59.63	62.66	37.08	35.20	36.12	91.25	91.71	91.48
	DP-Dep	67.28	59.75	63.28	37.87	33.69	35.63	91.54	91.74	91.64
	GP-Dep‡	69.25	58.99	63.70	38.88	36.11	37.43	92.37	92.05	92.21
Unsupervised	Attn	**68.41**	60.27	**64.08**	38.41	**36.58**	**37.45**	91.46	**92.10**	91.78
	Rollout	67.67	60.15	63.68	**39.13**	35.59	37.25	91.78	91.92	**91.85**
	Flow	66.71	**60.64**	63.53	38.85	35.98	37.35	**91.87**	91.80	91.84
	Attn†	66.94	59.62	63.06	37.91	35.72	36.75	91.68	91.50	91.59
	Rollout†	66.76	59.59	62.96	37.87	35.45	36.60	91.74	91.48	91.61
	Flow†	66.72	59.37	62.82	38.22	35.17	36.60	91.75	91.55	91.65
	PMask	67.14	59.28	63.14	37.76	35.98	36.84	91.67	91.92	91.80
	Grad	66.79	58.68	62.47	37.04	35.45	36.22	91.78	91.88	91.83
	GInput	66.33	59.69	62.81	38.13	35.31	36.66	91.72	91.94	91.83
	GInteg	67.31	59.57	63.21	38.09	35.09	36.53	91.86	91.61	91.73
Noise	Random	67.02	59.08	62.78	37.91	35.38	36.58	91.57	91.54	91.61

Table 5: The overall relation extraction results of four types of structures. **Baseline** indicates that no additional structure is added. **Bold** marks the highest in each column of a block excluding the "cheating " structure GP-Dep. The ‡ marker indicates "cheating" dependency parsing, which is not comparable with other structures. The † marker indicates the average of all layers, rather than the best layer.

Type	Structure	ACE05			SciERC			WebNLG		
		P	R	F1	P	R	F1	P	R	F1
Baseline	-	87.74	88.28	88.00	68.75	69.43	69.09	97.43	98.19	97.81
Light	Self-Loop	87.73	88.19	87.96	69.82	69.10	69.46	97.43	98.17	97.8
	Linear	87.81	88.42	88.11	69.32	69.06	69.18	97.49	98.25	97.87
Dependency	Dep	87.60	88.09	87.84	69.34	68.56	68.95	97.3	98.37	97.83
	DP-Dep	87.76	89.17	88.20	69.50	68.05	68.76	97.34	98.28	97.81
	GP-Dep‡	**88.40**	**88.56**	**88.48**	71.77	70.69	71.22	**98.47**	**98.80**	**98.63**
Unsupervised	Attn	87.61	88.35	87.98	69.51	**69.43**	69.47	97.27	**98.40**	97.83
	Rollout	87.81	87.96	87.87	**70.16**	69.41	**69.78**	97.45	98.32	97.88
	Flow	**88.07**	**88.42**	**88.25**	69.62	68.93	69.27	97.43	**98.40**	**97.92**
	Attn†	87.65	88.06	87.86	69.10	69.42	69.16	97.39	98.16	97.77
	Rollout†	87.71	88.06	87.88	68.80	69.15	69.45	97.43	98.18	97.8
	Flow†	87.74	88.10	87.92	68.95	69.15	69.27	97.38	98.22	97.80
	PMask	87.49	88.12	87.80	69.75	69.39	69.57	97.39	98.28	97.83
	Grad	87.74	88.18	87.96	68.63	67.97	68.30	97.21	98.26	97.73
	GInput	87.63	88.16	87.89	69.48	69.28	69.38	97.28	98.19	97.73
	GInteg	87.66	88.25	87.96	69.15	69.18	69.17	**97.50**	98.23	97.87
Noise	Random	87.73	88.18	87.96	69.28	69.00	69.14	97.34	98.22	97.78

Table 6: The overall entity extraction results of four types of structures. **Baseline** indicates that no additional structure is added. **Bold** marks the highest in each column of a block excluding the "cheating " structure GP-Dep. The ‡ marker indicates "cheating" dependency parsing, which is not comparable with other structures. The † marker indicates the average of all layers, rather than the best layer.