RelU-Net: Syntax-aware Graph U-Net for Relational Triple Extraction

Yunqi Zhang[1], **Yubo Chen**[1]*, **Yongfeng Huang**[1,2]
[1] Department of Electronic Engineering & BNRist, Tsinghua University, Beijing, China
[2] Zhongguancun Laboratory, Beijing, China
yq-zhang19@mails.tsinghua.edu.cn ybch14@gmail.com
yfhuang@tsinghua.edu.cn

Abstract

Relational triple extraction is a critical task for natural language processing. Existing methods mainly focused on capturing semantic information, but suffered from ignoring the syntactic structure of the sentence, which is proved in the relation classification task to contain rich relational information. This is due to the absence of entity locations, which is the prerequisite for pruning noisy edges from the dependency tree, when extracting relational triples. In this paper, we propose a unified framework to tackle this challenge and incorporate syntactic information for relational triple extraction. First, we propose to automatically contract the dependency tree into a core relational topology and eliminate redundant information with graph pooling operations. Then, we propose a symmetrical expanding path with graph unpooling operations to fuse the contracted core syntactic interactions with the original sentence context. We also propose a bipartite graph matching objective function to capture the reflections between the core topology and golden relational facts. Since our model shares similar contracting and expanding paths with encoder-decoder models like U-Net, we name our model as Relation U-Net (RelU-Net). We conduct experiments on several datasets and the results prove the effectiveness of our method.

1 Introduction

Relational Triple Extraction (RTE) is defined as automatically recognizing entity pairs and the semantic relations in the form of (*subject*, *relation*, *object*) from unstructured text. It is a critical task for natural language processing, especially for constructing large-scale Knowledge Graphs (KGs) from unlabeled corpus (Dong et al., 2014).

Recent work proposed several neural network methods to extract relational triples. For example, Zheng et al. (2017) formulated this task as sequence tagging problems but failed to extract overlapping triples. Wei et al. (2020) proposed a cascade tagging framework to solve the overlapping problem. Chen et al.(2021) proposed to extract the implicit relational triples which are not explicitly expressed in the sentence with a binary pointer network.

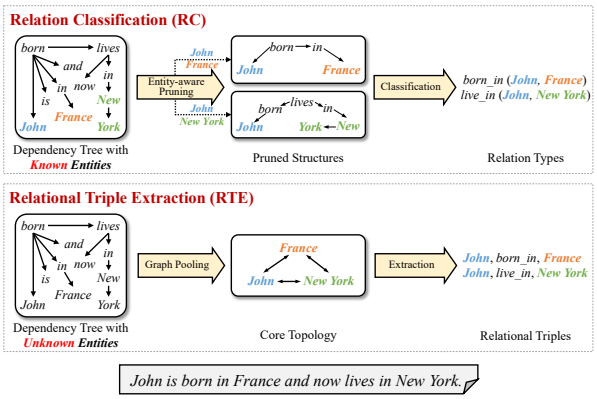

Figure 1: An example of the dependency-based relation classification method and our syntax-aware relational triple extraction method. The dashed arrows indicate the location injection of the known entities when pruning the dependency tree.

Existing methods achieved considerable success in capturing semantic information from relational mentions. However, they usually failed to incorporate syntactic structures of the sentence, which is proved to contain rich relational information in the Relation Classification (RC) task (Zhang et al., 2018; Guo et al., 2019). These syntax-aware RC methods usually pruned irrelevant dependency edges according to the known locations of the entity pair to eliminate the noise of the dependency tree. For example, the relational facts in Figure 1 (top) can be easily inferred given the dependency paths, which are pruned from the original dependency tree using the locations of the known entity pairs. Unfortunately, the locations of entities are unknown in the RTE task, as shown in Figure 1 (bottom). This absence makes it difficult to prune

*Corresponding author.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 4208 - 4217
December 7-11, 2022

dependency noise thus leads to an insufficient exploration of syntactic and relational information.

In this paper, we propose a unified framework to tackle this challenge and incorporate syntactic information for relational triple extraction. First, we propose to reduce the dependency tree and eliminate syntactic noise with graph pooling operations (Figure 1). We utilize MinCut pooling (Bianchi et al., 2020) to cluster similar nodes (words) and hence obtain a core relational topology. Next, we apply the inductive GraphSAGE algorithm (Hamilton et al., 2017) to propagate cluster information and capture the syntactic interactions underlying the core topology. Then, we propose to symmetrically expand the core topology with graph unpooling operations to integrate the syntactic interactions with the original sentence context. Finally, we propose a bipartite graph matching loss to induce the connections in the core topology to reflect golden relational facts, which is similar to the reflections between the dependency paths and golden facts (e.g. Figure 1) in the RC task. Since our model shares similar contracting and expanding paths with encoder-decoder models like U-Net (Ronneberger et al., 2015), we name our model as Relation U-Net (RelU-Net).

The main contributions of this paper are:

- We propose a unified framework to incorporate syntactic structures of the sentence for relation triple extraction.
- To eliminate disturbance caused by irrelevant syntactic information, we propose to automatically contract the dependency tree into a core topology with graph pooling operations.
- To fuse the syntactic interactions underlying the core topology with the original sentence context, we propose a symmetrical expanding path with graph unpooling operations.
- To establish reflections between core topology connections and golden relational facts, we propose a bipartite graph matching loss.

2 Related Work

In the early work of relational triple extraction, the task is addressed in a pipelined manner (Zelenko et al., 2003; Zhou et al., 2005; Chan and Roth, 2011; Gormley et al., 2015). They recognized all entities in the sentence and classified the relations between pairs of extracted entities separately. However, the pipeline methods usually suffered from error propagation and failed to capture the interactions between the entities and relations.

To address these issues, end-to-end models for joint extraction of entities and relations have become the dominant paradigm of this task, including feature-based models (Yu and Lam, 2010; Li and Ji, 2014; Ren et al., 2017) and neural network-based models (Miwa and Sasaki, 2014; Gupta et al., 2016; Miwa and Bansal, 2016; Zheng et al., 2017). For example, Ren et al. (2017) proposed to detect relation mentions and their entity arguments with distant supervision. Gupta et al. (2016) proposed to model the inter-dependencies of entities and relations through the entity-relation table proposed by (Miwa and Sasaki, 2014). Zheng et al. (2017) first formulated this task as a sequence tagging problem, but they failed to extract overlapping triples.

More recent work developed several strategies to address the overlapping triple problem, including sequence tagging based models (Wei et al., 2020; Wang et al., 2020; Zheng et al., 2021; Chen et al., 2021) and triple generation based models (Zeng et al., 2018, 2019, 2020; Sui et al., 2020; Huguet Cabot and Navigli, 2021). For example, Wei et al. (2020) proposed a cascade tagging scheme to simultaneously identify all possible overlapping triples. Chen et al. (2021) proposed a binary pointer network to extract overlapping relational triples and introduced a relational network to capture relational reasoning patterns for this task. Zeng et al. (2018) proposed a sequence-to-sequence triple generation model with copy mechanism, while Sui et al. (2020) proposed to treat this task as a direct set prediction problem. However, these methods mainly focused on learning semantic information of the sentence and usually suffered from ignoring syntactic patterns of the sentence. Although Fu et al. (2019) applied the dependency tree to extract regional features of the text, they did not prune irrelevant contents from the dependency tree, which may be sub-optimal.

Different from previous work, we propose to remove redundant information from the dependency tree with graph pooling operations and integrate core syntactic connections with the original sentence context with graph unpooling operations. We also propose a bipartite matching loss to guide the core syntactic connections to reflect golden relational facts, like in the RC task. Experimental results on several benchmark datasets prove the effectiveness of our method.

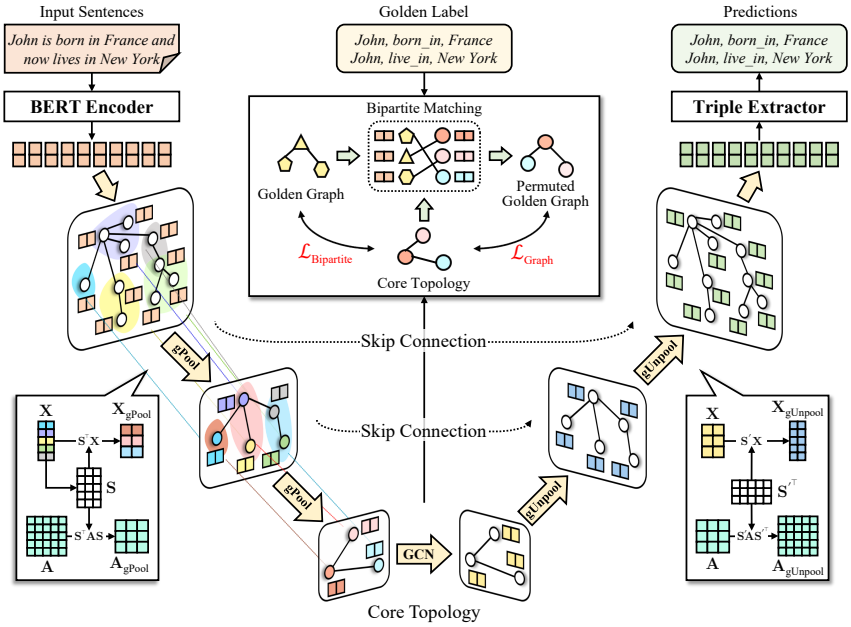

Figure 2: The overall framework of our approach.

3 Our Approach

The overall framework of our approach is illustrated in Figure 2. We introduce the Relation U-Net, the triple extractor and the details of model training in Section 3.1, 3.2 and 3.3, respectively.

3.1 Relation U-Net (RelU-Net)

The syntactic structure of the sentence has already been proved to contain rich relational information by existing RC methods (Zhang et al., 2018; Guo et al., 2019; Yu et al., 2020). They usually pruned irrelevant contents from the dependency tree to eliminate noise according to the known locations of each entity pair (Figure 1). However, the locations of entities are unknown in RTE, which makes it challenging to prune noisy dependency edges and sufficiently exploit syntactic information. To tackle this challenge, we first down-sample the dependency tree to summarize informative structures and reduce noise. The down-sampled core topology contains relation-relevant syntactic information of the sentence. Then, we symmetrically up-sample the core topology to enable precise triple localization. Our model has a contracting and expanding path of the dependency tree, which shares similar architecture with encoder-decoder models like U-Net (Ronneberger et al., 2015).

Therefore, we propose Relation U-Net, named as RelU-Net, to incorporate syntactic information for RTE. We first propose to automatically reduce the dependency tree into a core relational topology with graph pooling operations (Section 3.1.1). Then, we adopt inductive graph convolutions to capture the syntactic interactions underlying the core topology (Section 3.1.2). Finally, we propose to expand the core topology with graph unpooling operations for the fusion of the syntactic interactions and the semantic context (Section 3.1.3).

3.1.1 Graph Pooling

Given the input sentence $\{w_1, \ldots, w_n\}$ and its dependency tree, we first convert the words (nodes) into contextual representations with a text encoder such as BERT (Devlin et al., 2019), whose output denoted as $\mathbf{E} = [\mathbf{E}_1, \ldots, \mathbf{E}_n]$. Then, we represent the dependency tree with an adjacency matrix $\mathbf{A}_{\text{Dep}}$, where $(\mathbf{A}_{\text{Dep}})_{i,j} = 1$ if there exists a dependency edge between the i-th and the j-th word otherwise 0[1]. Next, we utilize the spectral-clustering MinCut pooling (Bianchi et al., 2020) operations to aggregate nodes with strong syntactic connections and similar semantic features. By clustering dependency nodes, we merge all irrelevant edges into the supernodes, guiding our model to focus more on critical syntactic interactions between the supernodes, thus reducing the negative impact of noisy contents. Specifically, MinCut pooling (denoted as gPool) captures syntactic interactions with a Graph Convolution Network (GCN, Section 3.1.2) and

[1]Following Zhang et al. (2018), we do not use the dependency types due to their limited discriminative power.

uses a Multi-Layer Perceptron (MLP) to learn a cluster assignment matrix $\mathbf{S} \in \mathbb{R}^{N \times C}$:

$$\mathbf{S} = \underset{N}{\text{softmax}}(\text{MLP}(\text{GCN}(\mathbf{X}, \mathbf{A}); \mathbf{\Theta})) \quad (1)$$

where N, C are numbers of input nodes and output clusters, $\mathbf{X} \in \mathbb{R}^{N \times d}, \mathbf{A} \in \mathbb{R}^{N \times N}$ are the input node features and adjacency matrix, and the softmax operation is to normalize the contributions of all input nodes to each output cluster. Then we compute the pooled node features $\mathbf{X}_{\text{gPool}}$ and adjacency matrix $\mathbf{A}_{\text{gPool}}$:

$$\mathbf{X}_{\text{gPool}} = \mathbf{S}^\top \mathbf{X}, \ \mathbf{A}_{\text{gPool}} = \mathbf{S}^\top \mathbf{A} \mathbf{S} \quad (2)$$

To emphasize the inter-supernode connections, we zero the diagonal and apply degree normalization to the pooled adjacency matrix:

$$\begin{aligned} \hat{\mathbf{A}}_{\text{gPool}} &= \mathbf{A}_{\text{gPool}} - \mathbf{I}_K \text{diag}(\mathbf{A}_{\text{gPool}}) \\ \widetilde{\mathbf{A}}_{\text{gPool}} &= \hat{\mathbf{D}}^{-\frac{1}{2}} \hat{\mathbf{A}}_{\text{gPool}} \hat{\mathbf{D}}^{-\frac{1}{2}} \end{aligned} \quad (3)$$

where $\hat{\mathbf{D}}$ is the degree matrix of $\hat{\mathbf{A}}_{\text{gPool}}$. Since these pooling operations are fully differentiable, the MinCut layer can be stacked multiple times:

$$\mathbf{X}^{(l+1)} = \text{ReLU}(\mathbf{X}^{(l)}_{\text{gPool}}), \ \mathbf{A}^{(l+1)} = \widetilde{\mathbf{A}}^{(l)}_{\text{gPool}} \quad (4)$$

where $\mathbf{X}^{(1)} = \mathbf{E}$, $\mathbf{A}^{(1)} = \mathbf{A}_{\text{Dep}}$. We perform L times pooling on the raw dependency tree and obtain a core topology $\mathcal{G}_{\text{Core}} = \{\mathbf{X}^{(L)}, \mathbf{A}^{(L)}\}$, as shown in Figure 2 (left).

Note that our model is similar to the multi-layer graph pooling proposed by Gao and Ji (2019), but we do not use their top-K graph pooling. This is because the top-K pooling reduces graph size by simply removing nodes, but the RTE task involves node merging, such as multi-word entities.

3.1.2 Graph Convolution

We apply multi-layer graph convolutions to capture the syntactic interactions underlying the core topology. However, this pooled topology changes dynamically with the parameter updates during training, thus making the conventional transductive graph convolution methods (Kipf and Welling, 2016) not suitable to our model. Therefore, we adopt the inductive GraphSAGE (Hamilton et al., 2017) algorithm, which first uniformly samples neighbor nodes w.r.t. each node and then aggregates the features of the sampled nodes. Formally, the convolution of the p-th layer is formulated as:

$$\mathbf{G}_i^{(p+1)} = \text{ReLU}\left(\mathbf{W}^{(p)}[\mathbf{G}_i^{(p)}; \text{Ave}(\mathbf{G}_{\mathcal{S}_i}^{(p)})]\right) \quad (5)$$

where $\mathbf{W}^{(p)}$ stands for the parameters of the p-th layer, $\mathcal{S}_i$ denotes the i-th node's sampled neighbors in the topology, and $\mathbf{G}^{(1)} = \mathbf{X}^{(L)}$. We also use GraphSAGE in the GCN of graph pooling (Equation 1) because the intermediate graphs of the pooling path are also dynamic.

3.1.3 Graph Unpooling

To fuse the syntactic representations of the core topology with the original sentence context, we propose a reverse expanding path with graph unpooling operations to enable precise triple localization, as illustrated in Figure 2 (right). During graph pooling process, apart from learning the cluster assignment matrix $\mathbf{S}$, we also learn a cluster decomposition matrix $\mathbf{S}^{'}$ with another MLP: $\mathbf{S}^{'} = \text{softmax}(\text{MLP}(\text{GCN}(\mathbf{X}, \mathbf{A}); \mathbf{\Theta}^{'}))$. Then, we transpose Equation 2 to achieve graph unpooling:

$$\mathbf{X}_{\text{gUnpool}} = \mathbf{S}^{'} \mathbf{X}, \ \mathbf{A}_{\text{gUnpool}} = \mathbf{S}^{'} \mathbf{A} \mathbf{S}^{'\top} \quad (6)$$

We also add symmetrical skip-connections between the corresponding pooling and unpooling layers to combine semantic and syntactic information at different levels (Figure 2).

3.2 Relational Triple Extractor

We adopt CASREL (Wei et al., 2020) to the output representations $\mathbf{H}$ of the RelU-Net to extract relational triples, which consists of a binary subject tagger and a set of relational-specific object taggers. The subject tagger predicts the start and end positions of all possible subjects with two binary classifiers: $\mathbf{p}^s_{s/e} = \sigma\big(\mathbf{W}^s_{s/e}\mathbf{H} + \mathbf{b}^s_{s/e}\big)$ where σ is the sigmoid function, $\mathbf{W}^s_{s/e}$ and $\mathbf{b}^s_{s/e}$ are the parameters of the subject classifiers. Then, each relational-specific object tagger takes the averaged representation of the k-th subject's start and end tokens as $\mathbf{s}_k$ and predicts the start and end positions of all objects corresponding to the current subject under the r-th relation: $\mathbf{p}^{o,r}_{s/e,k} = \sigma\left(\mathbf{W}^{o,r}_{s/e,k}(\mathbf{H} + \mathbf{s}_k) + \mathbf{b}^{or}_{s/e}\right)$ where $\mathbf{W}^{or}_{s/e}$ and $\mathbf{b}^{or}_{s/e}$ are the parameters of the r-th relation-specific object classifiers. We set the binary tags to 1 if their probabilities exceed some threshold (0.5 in our model) otherwise 0. Then we extract the subjects and objects by matching the nearest start-end position pair. We refer readers to (Wei et al., 2020) for more detailed descriptions.

3.3 Model Training

We calculate a Binary Cross-Entropy (BCE) $f(\mathbf{y}, \mathbf{p}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log p_i + (1 - y_i) \log(1 - p_i)$

as the loss of a triple extractor's predictions:

$$\mathcal{L}_t = \sum_{*\in\{s,e\}} (f(\mathbf{y}^s_*, \mathbf{p}^s_*) + \sum_{r,k} f(\mathbf{y}^{o,r}_{*,k}, \mathbf{p}^{o,r}_{*,k})), \quad (7)$$

where $\mathbf{y}$ are the binary labels corresponding to the position probabilities $\mathbf{p}$.

In addition, we observe from the RC task that a dependency path connecting two entities reflects a relational fact (Figure 1). Similarly, we expect the connections in the core topology to have reflections to the golden relational facts. Therefore, we propose an objective function that minimizes the adjacency similarity between the core topology and the golden relational triples. We represent the golden triples with a relational graph $\mathcal{G}_{\text{Gold}} = \{\mathbf{X}^G, \mathbf{A}^G\}$, whose nodes are subject and object entities, $\mathbf{X}^G$ is computed by averaging the word representations $\mathbf{E}$ corresponding to the nodes' entities, and $\mathbf{A}^G_{i,j} = 1$ if there exists a golden triple between the i-th and the j-th node otherwise 0. However, the node orders of $\mathcal{G}_{\text{Core}}$ and $\mathcal{G}_{\text{Gold}}$ may mismatch, making it difficult to directly compare their adjacency matrices because they are sensitive to node permutations.

To address this issue, we propose to find the optimal bipartite matching between two sets of graph nodes and compute the similarity score between the core topology and the *permuted* golden graph, as illustrated in Figure 2 (top). We formulate the optimal bipartite matching as a permutation π^* with the minimum cost: $\pi^* = \underset{\pi\in\Pi_m}{\operatorname{argmin}} \sum_{i=1}^{m} \mathcal{C}(\mathbf{X}^G_{\pi_i}, \mathbf{X}^{(L)}_i)$, where Π_m is the permutation space with length m. $\mathcal{C}$ is a pair-wise matching cost between the two sets of graph nodes and is defined as a bilinear score:

$$\mathcal{C}^\pi_i \coloneqq \mathcal{C}(\mathbf{X}^G_{\pi_i}, \mathbf{X}^{(L)}_i) = -\mathbf{X}^{G\top}_{\pi_i} \mathbf{W}_m \mathbf{X}^{(L)}_i \quad (8)$$

where $\mathbf{W}_m$ are learnable parameters of the cost function. The optimal permutation π^* is computed via the Hungarian algorithm[2] in polynomial time. Moreover, we add a bipartite loss to minimize the approximate[3] lower bound of the matching cost:

$$\begin{aligned}\mathcal{L}_{\text{Bipartite}} &= \sum_{i=1}^{m} \log\Big(\underset{i}{\text{softmax}}\, \mathcal{C}^{\pi^*}_i\Big) \\ &= \sum_{i=1}^{m} \mathcal{C}^{\pi^*}_i - m \cdot \log \sum_{j=1}^{m} \exp \mathcal{C}^{\pi^*}_j\end{aligned} \quad (9)$$

Finally, we permute the golden graph's adjacency $\mathbf{A}^G$ with the optimal matching π^* and compute its similarity to the core topology's adjacency $\mathbf{A}^{(L)}$ with a point-wise BCE:

$$\mathcal{L}_{\text{Graph}} = \text{BCE}(\mathbf{A}^G_{\pi^*}, \mathbf{A}^{(L)}). \quad (10)$$

We train our RelU-Net with the joint loss $\mathcal{L} = \mathcal{L}_t + \alpha\mathcal{L}_{\text{Bipartite}} + \beta\mathcal{L}_{\text{Graph}}$, in which α and β are hyper-parameters.

4 Experiments

4.1 Datasets and Evaluation Metrics

We evaluate our method on two widely-used benchmark datasets: NYT (Riedel et al., 2010) and WebNLG (Gardent et al., 2017). NYT consists of sentences sampled from New York Times news articles and contains 24 relation types. WebNLG was originally proposed for natural language generation and first introduced in the RTE task by Zeng et al. (2018), which contains 171 relation types.

Following previous work (Zeng et al., 2018; Wei et al., 2020; Chen et al., 2021), we divide the sentences into three classes: *Normal*, *EntitypairOverlap* (EPO) and *SingleEntityOverlap* (SEO) according to different overlapping patterns of triples, as shown in Table 1. We adopt the same partial match score following previous work (Zheng et al., 2017; Wei et al., 2020; Chen et al., 2021) for evaluation. We regard the extracted triple as correct if and only if the relation and the heads of subject and object are all correct. We report the standard micro precision, recall, and F_1 scores on both datasets.

Dataset	NYT		WebNLG	
	Train	Test	Train	Test
Normal	37013	3266	1596	246
SEO	9782	1297	227	457
EPO	14735	978	3406	26
ALL	56195	5000	5019	703

Table 1: Statistics of NYT and WebNLG datasets.

4.2 Experimental Settings

The hyper-parameters are determined on the validation set. We use the spaCy toolkit[4] to parse the dependencies of sentences. We adopt $\text{BERT}_{\text{BASE}}$ and $\text{RoBERTa}_{\text{LARGE}}$ as our text encoders following (Chen et al., 2021). We use 2 layers of graph pooling, graph unpooling, and GraphSAGE convolutions in our model. The output node numbers of

[2]https://en.wikipedia.org/wiki/Hungarian_algorithm

[3]The approximation is to ensure that the loss is positive.

[4]https://github.com/explosion/spaCy

Method	NYT			WebNLG		
	Prec.	Rec.	F_1	Prec.	Rec.	F_1
CASREL$_{Random}$ (Wei et al., 2020)	81.5	75.7	78.5	84.7	79.5	82.0
CASREL$_{BERT}$ (Wei et al., 2020)	89.7	89.5	89.6	93.4	90.1	91.7
TPLinker$_{BERT}$ (Wang et al., 2020)	91.3	92.5	91.9	91.8	92.0	91.9
SPN$_{BERT}$ (Sui et al., 2020)	93.3	91.7	92.5	93.1	93.6	93.4
CGT$_{Random}$ (Ye et al., 2021)	90.8	77.7	83.7	87.6	70.5	78.1
CGT$_{UniLM}$ (Ye et al., 2021)	**94.7**	84.2	89.1	92.9	75.6	83.4
PFN$_{BERT}$ (Yan et al., 2021)	-	-	92.4	-	-	93.6
TDEER$_{BERT}$ (Li et al., 2021)	93.0	92.1	92.5	93.8	92.4	93.1
PRGC$_{Random}$ (Zheng et al., 2021)	89.6	82.3	85.8	90.6	88.5	89.5
PRGC$_{BERT}$ (Zheng et al., 2021)	93.3	91.9	92.6	94.0	92.1	93.0
†R-BPtrNet$_{BERT}$ (Chen et al., 2021)	92.7	92.5	92.6	93.7	92.8	93.3
‡R-BPtrNet$_{BERT}$ (Chen et al., 2021)	92.7	91.6	92.1	93.6	92.9	93.3
†R-BPtrNet$_{RoBERTa}$ (Chen et al., 2021)	94.0	92.9	<u>93.5</u>	94.3	93.3	93.8
‡R-BPtrNet$_{RoBERTa}$ (Chen et al., 2021)	93.3	<u>93.0</u>	93.2	94.2	93.2	93.7
RelU-Net$_{Raw}$	87.9	87.3	87.6	90.3	89.1	89.7
RelU-Net$_{Random}$	89.4	87.6	88.5	93.4	88.9	91.1
⋆RelU-Net$_{BERT}$	93.3	92.9	93.1	<u>94.9</u>	<u>93.7</u>	<u>94.3</u>
⋆RelU-Net$_{RoBERTa}$	<u>94.2</u>	**93.3**	**93.7**	**95.4**	**94.4**	**94.9**

Table 2: Performance of RelU-Net and previous state-of-the-art models on the NYT and WebNLG test sets. The best scores are in bold and the second-best scores are underlined. † marks models using entity type information on the NYT dataset and ‡ marks models without entity types re-implemented by ours for a fair comparison. ⋆ denotes significant improvements over the corresponding R-BPtrNet models with $p < 0.001$ under a t-test. RelU-Net$_{Raw}$ removes the graph pooling and unpooling operations from RelU-Net$_{BERT}$, i.e. utilizing the full dependency tree. RelU-Net$_{Random}$ is the model with the same transformer encoder as BERT$_{\text{BASE}}$ but initialized with random numbers rather than the pre-trained weights.

the graph pooling layers are set to 16 and 8, respectively. The weights α and β of bipartite matching loss and graph similarity loss are set to 1.0 and 0.1, respectively. We use Adam optimizer (Kingma and Ba, 2014) to fine-tune the pre-trained BERT weights with the learning rate of 10^{-5} and train other parameters with the learning rate of 5×10^{-4}. We train our model for 200 epochs with batch size as 10 on both datasets. We choose the model with the best validation performance and report the scores on the test set.

4.3 Performance Evaluation

We report the evaluation results on the NYT and WebNLG test sets in Table 2. We also compare our RelU-Net with several previous state-of-the-art models: (1) **CASREL** (Wei et al., 2020) proposed a cascade binary tagging framework with a subject tagger and a set of relational-specific object taggers. (2) **TPLinker** (Wang et al., 2020) proposed a novel handshaking tagging scheme to link token pairs. (3) **SPN** (Sui et al., 2020) proposed a relational triple set prediction network with non-autoregressive transformers. (4) **CGT** (Ye et al., 2021) proposed a novel generative transformer for contrastive triple extraction. (5) **PFN** (Yan et al., 2021) proposed a partition filter network to model two-way interaction between NER and RE. (6) **TDEER** (Li et al., 2021) proposed a translating decoding schema with negative samples. (7) **PRGC** (Zheng et al., 2021) proposed to model potential relation and global correspondence through decomposing this task into three subtasks. (8) **R-BPtrNet** (Chen et al., 2021) proposed a reasoning pattern enhanced binary pointer network to extract implicit relational triples.

From Table 2 we have several observations. First, the performance of RelU-Net$_{Raw}$ significantly drops on both datasets compared with CASREL$_{BERT}$. It indicates that the full dependency tree contains much noise and will greatly hurt the model performance if used without pruning.

Method	NYT								WebNLG							
	Nor.	SEO	EPO	N=1	N=2	N=3	N=4	N≥5	Nor.	SEO	EPO	N=1	N=2	N=3	N=4	N≥5
CASREL$_{BERT}$	87.3	91.4	92.0	88.2	90.3	91.9	94.2	83.7	89.4	92.2	94.7	89.3	90.8	94.2	92.4	90.9
TPLinker$_{BERT}$	90.1	93.4	94.0	90.0	92.9	93.1	96.1	90.0	87.9	92.5	95.3	88.0	90.1	94.6	93.3	91.6
SPN$_{BERT}$	90.8	94.0	94.1	90.9	93.4	94.2	95.5	90.6	-	-	-	-	-	-	-	-
PRGC$_{BERT}$	91.0	94.0	94.5	**91.1**	93.0	93.5	95.5	93.0	90.4	93.6	95.9	89.9	91.6	95.0	94.8	92.8
R-BPtrNet$_{BERT}$	90.4	**94.4**	95.2	89.5	93.1	93.5	**96.7**	91.3	89.5	93.9	96.1	88.5	91.4	96.2	94.9	94.2
RelU-Net$_{BERT}$	**91.3**	94.1	**95.3**	90.9	**93.5**	**94.3**	96.2	**93.3**	**90.6**	**94.5**	**97.0**	**90.1**	**93.1**	**96.4**	**96.1**	**94.5**

Table 3: F_1 scores on sentences with different overlapping patterns and different triple numbers. The best scores are in bold and the second-best scores are underlined. N stands for the number of triples in the sentence.

Second, RelU-Net$_{Random}$ significantly outperforms other randomly initialized models on the NYT and WebNLG datasets. Moreover, RelU-Net$_{Random}$ even outperforms RelU-Net$_{Raw}$ where the latter uses pre-trained BERT weights while the former does not. It indicates the effectiveness of our method to incorporate the syntactic structure of the sentence by automatically contracting the dependency tree and reducing noisy contents. Finally, RelU-Net$_{BERT}$ and RelU-Net$_{RoBERTa}$ further outperforms RelU-Net$_{Random}$ and other baseline models. It indicates that the pre-trained language models bring more prior knowledge from unlabeled corpus and enhance the model performance.

4.4 Performance on Different Sentence Types

Following previous work (Wei et al., 2020; Zheng et al., 2021; Chen et al., 2021), we divide the test sets of two datasets according to the overlapping patterns and the triple counts to investigate the ability of our model in handling complex sentences. The results are shown in Table 3. We observe that our RelU-Net model outperforms the baseline models in almost all the subsets, especially on the WebNLG dataset. We speculate that this is because we train the model with the graph similarity loss (Section 3.3) to bootstrap the core topology connections to reflect relational facts, thus the complex interactions between multiple relational triples can be naturally captured through the graph convolutions over the core topology. In general, the results on different sentence types show the effectiveness of our model in complicated scenarios.

4.5 Ablation Study

In this section, we conduct an ablation study on the NYT test set to demonstrate the contribution of each component of our model. The result is shown in Table 4. Note that when removing GraphSAGE, we use the transductive convolution (Kipf and Welling, 2016) instead. From Table 4, we observe that the GraphSAGE convolution constitutes a significant contribution to the model performance. It indicates that the inductive convolution can better fit the dynamic core topology and intermediate graphs in the pooling path and improve the ability of capturing syntactic interactions. The bipartite matching loss $\mathcal{L}_{\text{Bipartite}}$ also brings improvements

Method	Prec.	Rec.	F_1
RelU-Net$_{BERT}$	**93.3**	**92.9**	**93.1**
w/o GraphSAGE	92.2	91.0	91.6
w/o $\mathcal{L}_{\text{Bipartite}}$	93.1	91.5	92.3
w/o $\mathcal{L}_{\text{Bipartite}}$+$\mathcal{L}_{\text{Graph}}$	91.8	90.6	91.2
w/o All	91.2	90.0	90.6

Table 4: An ablation study of the RelU-Net$_{BERT}$.

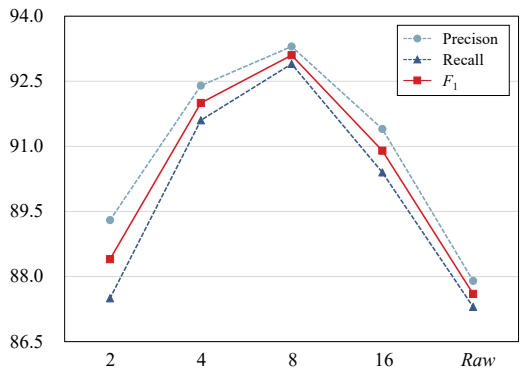

Figure 3: Performance of RelU-Net$_{BERT}$ on the NYT test set with different core topology sizes. In each setting, the node count of the intermediate pooled graph equals to the double of the core topology size. "*Raw*" stands for using the full dependency tree without any pooling or unpooling operations.

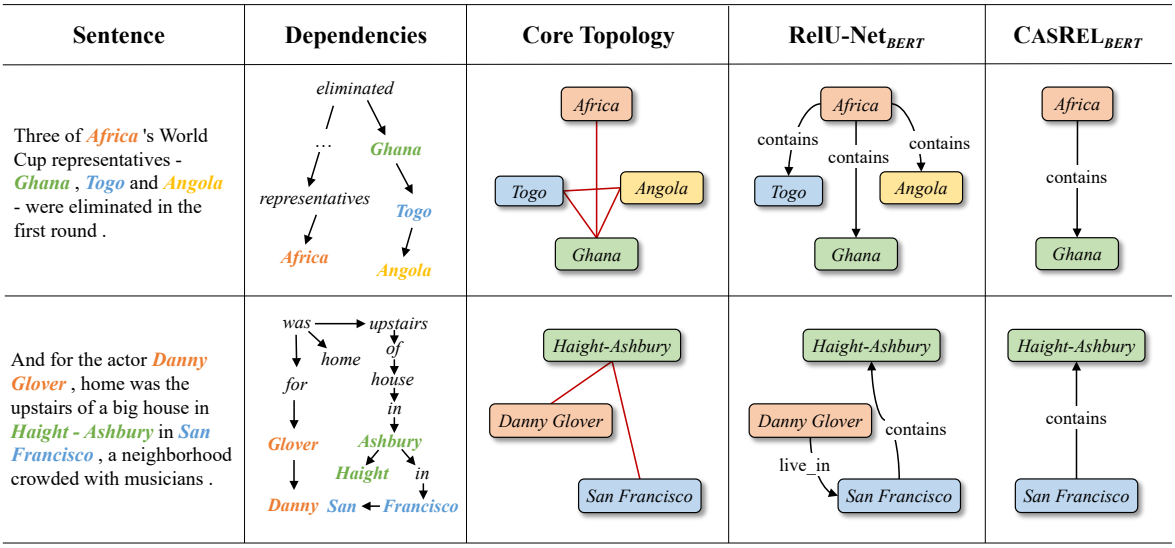

Figure 4: Examples of sentences and the corresponding predictions from the RelU-Net$_{BERT}$ and CASREL$_{BERT}$ models. We present the *inversely permuted* core topology to better show the reflections of its connections to relational facts. Difference entities are distinguished with different colors. Deep red connections represent those whose corresponding values in the adjacency matrix of the core topology are greater than 0.5.

to the model because it helps minimize the approximate lower bound of the node matching cost, which is approximately equivalent to minimizing the actual cost with a relaxation since the real optimal permutation is unknown. Moreover, removing the bipartite matching and graph similarity loss together causes a significant performance drop. It demonstrates the effectiveness of our training objectives in capturing the reflections between syntactic connections and relational facts. Finally, the model without all three components produces the worst performance, which proves that our model can sufficiently explore syntactic information and improve RTE performance.

4.6 Influence of Core Topology Size

To investigate the balance between pruning irrelevant contents and preserving informative edges, we conduct experiments on the NYT test sets with different core topology sizes, which stand for different ratios of graph reduction. The results are illustrated in Figure 3. We observe that when the size is too small, the performance drops significantly. We hypothesize that too aggressive pooling causes undesired pruning of informative contents, or even merging of multiple entities, and hurts the performance. When the topology size increases up to the most extreme setting of no pooling, the noisy content cannot be sufficiently eliminated thus leads to performance degradation.

4.7 Case Study

Figure 4 shows the comparison of the RelU-Net$_{BERT}$ and CASREL$_{BERT}$ models on two examples. In the first example, the dependency tree itself contains conjunct patterns between the entities "*Ghana*", "*Togo*" and "*Angola*" and the patterns are successfully preserved in the core topology. In the second example, despite the redundancy of the dependency structure between the entities "*Danny Glover*" and "*San Francisco*", the graph pooling operations eliminate the irrelevant contents and the corresponding connection in the core topology is informative enough for the triple to be extracted. We can observe that our model sufficiently explores the syntactic information of the dependency trees, while CASREL only captures semantic information and thus yields worse predictions than our model. The above observations demonstrate the effectiveness of our method in incorporating syntactic structures for relational triple extraction.

5 Conclusion

In this paper, we propose a unified framework to incorporate syntactic structures of the sentence for relation triple extraction. We propose a graph pooling network to automatically prune the dependency tree to a core topology and remove useless information. We propose a symmetrical graph unpooling network to integrate the syntactic interactions underlying the core topology with the original sen-

tence context. We also propose a bipartite graph matching objective function to learn the reflections between the core syntactic interactions and golden relational facts. We conduct experiments on two benchmark datasets, and the results demonstrate the effectiveness of our method.

6 Limitations

There are two main drawbacks of our model. First, our model employs multiple graph pooling, unpooling, and convolution layers, resulting in high time complexity and high computational resource demand. It usually takes a longer time to achieve the best performance compared to other models. Second, the bipartite matching loss in Equation 9 is approximated. This may cause the values to enter the saturation zone of the softmax function, which will lead to the vanishing gradient problem. Therefore, the instability and the slow convergence during training are also limitations of our model.

Acknowledgement

We would like to thank the anonymous reviewers for their valuable comments on this paper. This work was supported by National Key Research and Development Program of China under Grant number 2021ZD0113902 and the National Natural Science Foundation of China under Grant numbers U1936216 and U1936208. We would like to thank the VMWare gift funding's partial support to the authors.

References

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2020. Spectral clustering with graph neural networks for graph pooling. In *ICML*, pages 874–883.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *ACL-HLT*, pages 551–560.

Yubo Chen, Yunqi Zhang, Changran Hu, and Yongfeng Huang. 2021. Jointly extracting explicit and implicit relational triples with reasoning pattern enhanced binary pointer network. In *NAACL-HLT*, pages 5694–5703.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, pages 601–610.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *ACL*, pages 1409–1418.

Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. In *ICML*, pages 2083–2092.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *ACL*, pages 179–188.

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *EMNLP*, pages 1774–1784.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *ACL*, pages 241–251.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *COLING*, pages 2537–2547.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*, volume 30. Curran Associates, Inc.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of EMNLP*, pages 2370–2381.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *ACL*, pages 402–412.

Xianming Li, Xiaotian Luo, Chenghao Dong, Daichuan Yang, Beidi Luan, and Zhen He. 2021. Tdeer: An efficient translating decoding schema for joint extraction of entities and relations. In *EMNLP*, pages 8055–8064.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *ACL*, pages 1105–1116.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *EMNLP*, pages 1858–1869.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *WWW*, pages 1015–1024.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer.

Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, Xiangrong Zeng, and Shengping Liu. 2020. Joint entity and relation extraction with set prediction networks. *arXiv preprint arXiv:2011.01675*.

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 *COLING*, pages 1572–1582.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *ACL*, pages 1476–1488.

Zhiheng Yan, Chong Zhang, Jinlan Fu, Qi Zhang, and Zhongyu Wei. 2021. A partition filter network for joint entity and relation extraction. In *EMNLP*, pages 185–197.

Hongbin Ye, Ningyu Zhang, Shumin Deng, Mosha Chen, Chuanqi Tan, Fei Huang, and Huajun Chen. 2021. Contrastive triple extraction with generative transformer. In *AAAI*, volume 35, pages 14257–14265.

Bowen Yu, Xue Mengge, Zhenyu Zhang, Tingwen Liu, Wang Yubin, and Bin Wang. 2020. Learning to prune dependency trees with rethinking for neural relation extraction. In *COLING*, pages 3842–3852.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *COLING 2010: Posters*, pages 1399–1407.

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

Daojian Zeng, Haoran Zhang, and Qianying Liu. 2020. Copymtl: Copy mechanism for joint extraction of entities and relations with multi-task learning. In *AAAI*, pages 9507–9514.

Xiangrong Zeng, Shizhu He, Daojian Zeng, Kang Liu, Shengping Liu, and Jun Zhao. 2019. Learning the extraction order of multiple relational facts in a sentence with reinforcement learning. In *EMNLP-IJCNLP*, pages 367–377.

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 *ACL*, pages 506–514.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*, pages 2205–2215.

Hengyi Zheng, Rui Wen, Xi Chen, Yifan Yang, Yunyan Zhang, Ziheng Zhang, Ningyu Zhang, Bin Qin, Xu Ming, and Yefeng Zheng. 2021. Prgc: Potential relation and global correspondence based joint relational triple extraction. In *ACL*, pages 6225–6235.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *ACL*, pages 1227–1236.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *ACL*, pages 427–434.