GraphRel: Modeling Text as Relational Graphs for Joint Entity and Relation Extraction

Tsu-Jui Fu
Academia Sinica
s103062110@m103.nthu.edu.tw

Peng-Hsuan Li
Academia Sinica
jacobvsdanniel@gmail.com

Wei-Yun Ma
Academia Sinica
ma@iis.sinica.edu.tw

Abstract

In this paper, we present **GraphRel**, an end-to-end relation extraction model which uses graph convolutional networks (GCNs) to jointly learn named entities and relations. In contrast to previous baselines, we consider the interaction between named entities and relations via a relation-weighted GCN to better extract relations. Linear and dependency structures are both used to extract both sequential and regional features of the text, and a complete word graph is further utilized to extract implicit features among all word pairs of the text. With the graph-based approach, the prediction for overlapping relations is substantially improved over previous sequential approaches. We evaluate GraphRel on two public datasets: NYT and WebNLG. Results show that GraphRel maintains high precision while increasing recall substantially. Also, GraphRel outperforms previous work by **3.2%** and **5.8%** (F1 score), achieving a new state-of-the-art for relation extraction.

1 Introduction

Extracting pairs of entity mentions with semantic relations, i.e., triplets such as (*BarackObama*, *PresidentOf*, *UnitedStates*), is a central task in information extraction and allows automatic knowledge construction from unstructured text. Though important and well-studied, three key aspects are yet to be fully handled in an unified framework:

- End-to-end joint modeling of entity recognition and relation extraction;
- Prediction of overlapping relations, i.e., relations that share a common mention;
- Consideration of the interaction between relations, especially overlapping relations.

Traditionally, a pipelined approach is used to first extract entity mentions using a named entity recognizer and then predict the relations between every pair of extracted entity mentions (Zelenko et al., 2003; Zhou et al., 2005; Chan and Roth, 2011). Joint entity recognition and relation extraction models (Yu and Lam, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014; Ren et al., 2017) have been built to take advantage of the close interaction between these two tasks. While showing the benefits of joint modeling, these complicated methods are feature-based structured learning systems and hence rely heavily on feature engineering.

With the success of deep neural networks, NN-based automatic feature learning methods have been applied to relation extraction. These methods use CNN, LSTM, or Tree-LSTM on the word sequence between two entity mentions (Zeng et al., 2014; dos Santos et al., 2015), the shortest dependency paths between two entity mentions (Yan et al., 2015; Li et al., 2015), or the minimal constituency sub-tree spanning two entity mentions (Socher et al., 2012) to encode relevant information for each pair of entity mentions. However, these methods are not end-to-end joint modeling of entities and relations. They assume entity mentions are given and are expected to degrade significantly in performance when a named entity recognizer is needed in the pipeline for real world usage.

Another challenge for relation extraction is how to take into account the interaction between relations, which is especially important for overlapping relations, i.e., relations sharing common entity mentions. For example, (*BarackObama*, *PresidentOf*, *UnitedStates*) can be inferred from (*BarackObama*, *Governance*, *UnitedStates*); the two triplets are said to exhibit *EntityPairOverlap*. Another case is that the former triplet could also be inferred from (*BarackObama*, *LiveIn*, *WhiteHouse*) and (*WhiteHouse*, *PresidentialPalace*, *UnitedStates*), where the latter two are said to exhibit *SingleEntityOverlap*. Although common in knowledge base completion, such interaction, whether via direct deduction or indirect

evidence, is particularly difficult for joint entity recognition and relation extraction models, where entities are not present in the input. Indeed, although Zheng et al. (2017) propose a strong neural end-to-end joint model of entities and relations based on an LSTM sequence tagger, they have to completely give up overlapping relations.

In this paper, we propose **GraphRel**, a neural end-to-end joint model for entity recognition and relation extraction that is the first to handle all three key aspects in relation extraction. GraphRel learns to automatically extract hidden features for each word by stacking a Bi-LSTM sentence encoder and a GCN (Kipf and Welling, 2017) dependency tree encoder. Then GraphRel tags entity mention words and predicts relation triplets that connect mentions, where is the 1st-phase prediction.

To gracefully predict relation triplets while taking into account the interactions between them, we add a novel 2nd-phase relation-weighted GCN to GraphRel. Already guided by both entity loss and relation loss, the 1st-phase GraphRel extracts node hidden features along dependency links while establishing a new fully connected graph with relation-weighted edges. Then, by operating on the intermediate graph, the 2nd-phase GCN effectively considers the interaction between entities and (possibly overlapping) relations before the final classification for each edge. With GraphRel, our contribution is threefold:

- Linear and dependency structures, as well as implicit features among all word pairs of the text, are considered by our method;
- We perform end-to-end, joint modeling of entities and relations while considering all word pairs for prediction;
- The interaction between entities and relations is carefully considered.

We evaluate the method on two public relation extraction datasets: NYT and WebNLG. The experimental result shows that GraphRel substantially improves the overlapping relations over previous work, and achieves a new state-of-the-art on both datasets.

2 Related Work

The BiLSTM-GCN encoder part of our model resembles the BiLSTM-TreeLSTM model proposed by Miwa and Bansal (2016), as they also stack a dependency tree on top of sequences to jointly model entities and relations. They use Bi-LSTM on each sentence for automatic feature learning, and the extracted hidden features are shared by a sequential entity tagger and a shortest dependency path relation classifier. However, while introducing shared parameters for joint entity recognition and relation extraction, they must still pipeline the entity mentions predicted by the tagger to form mention pairs for the relation classifier.

Instead of trying to classify each mention pair as in previous work, Zheng et al. (2017) formulate relation extraction as a sequential tagging problem (NovelTagging) as with entity recognition. This allows them to model relation extraction by an LSTM decoder on top of a Bi-LSTM encoder. However, while showing promising results on the NYT dataset, their strength comes from focusing on isolated relations and completely giving up overlapping relations, which are relatively rare in the dataset. In comparison, the proposed GraphRel gracefully handles all types of relations while being end-to-end and jointly modeling entity recognition.

Zeng et al. (2018) then propose an end-to-end sequence-to-sequence model for relation extraction. They encode each sentence by a Bi-LSTM, and use the last encoder hidden state to initialize one (OneDecoder) or multiple (MultiDecoder) LSTMs for dynamic decoding relation triplets. When decoding, triplets are generated by selecting a relation and copying two words from the sentence. The seq2seq setup partially handles interaction between triplets. However, interactions between relations are only unidirectionally captured by considering previous generated triplets with a compulsory linear order when generating a new one. Instead, in this paper, we propose propagating entity and relation information on a word graph with automatically learned linkage by applying 2nd-phase GCN on top of the LSTM-GCN encoder.

Recently, considering dependency structure by GCN has been used in many natural language processing (NLP) tasks. Marcheggiani and Titov (2017) applies GCN on word sequences for semantic role labeling. Liu et al. (2018) encode long documents via GCN to perform text matching. Cetoli et al. (2016) combine RNN and GCN to recognize named entities. There are also some works (Peng et al., 2017; Zhang et al., 2018; Qian et al.,

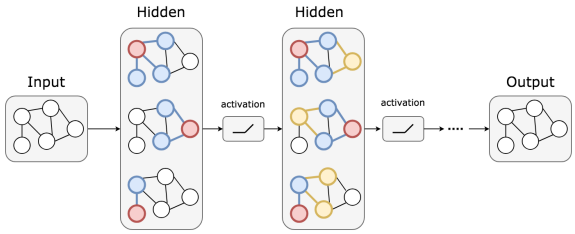

Figure 1: Graph Convolutional Network (GCN)

2019; Luan et al., 2019) about considering dependency structure of word sequence for relation extraction. In our proposed GrpahRel, we not only stack Bi-LSTM and GCN to consider both linear and dependency structures but also adopt a 2nd-phase relation-weighted GCN to further model the interaction between entities and relations.

3 Review of GCN

As convolutional neural network (CNN), Graph Convolutional Network (GCN) (Kipf and Welling, 2017) convolves the features of neighboring nodes and also propagates the information of a node to its nearest neighbors. Shown in Fig. 1, by stacking GCN layers, GCN can extract regional features for each node.

A GCN layer retrieves new node features by considering neighboring nodes' features with the following equation:

$$h_u^{l+1} = ReLU\left(\sum_{v \in N(u)} \left(W^l h_v^l + b^l\right)\right),$$

where u is the target node and $N(u)$ represents the neighborhood of u, including u itself; h_v^l denotes the hidden feature of node v at layer l; W and b are learnable weights, mapping the feature of a node onto adjacent nodes in the graph; and $h \in \mathbb{R}^f$, $W \in \mathbb{R}^{f \times f}$, and $b \in \mathbb{R}^f$, where f is the feature size.

4 Methodology

The overall architecture of the proposed GraphRel which contains 2 phases prediction is illustrated in Fig. 2. In the 1st-phase, we adopt bi-RNN and GCN to extract both sequential and regional dependency word features. Given the word features, we predict relations for each word pair and the entities for all words.

Then, in 2nd-phase, based on the predicted 1st-phase relations, we build complete relational graphs for each relation, to which we apply GCN on each graph to integrate each relation's information and further consider the interaction between entities and relations.

4.1 1st-phase Prediction

As the state-of-the-art text feature extractor (Marcheggiani and Titov, 2017; Cetoli et al., 2016), to take into account both sequential and regional dependencies, we first apply bi-directional RNN to extract sequential features and then use bi-directional GCN to further extract regional dependency features. Then, based on the extracted word features, we predict the relation for each word pair along with the word entity.

4.1.1 Bi-LSTM

We use the well-known long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) as our bi-RNN cell. For each word, we combine the word embedding and part-of-speech (POS) embedding as the initial feature:

$$h_u^0 = Word(u) \oplus POS(u),$$

where h_u^0 represents the initial feature of word u, and $Word(u)$ and $POS(u)$ are the word and POS embedding of word u, respectively. We use pre-trained word embeddings from GloVe (Pennington et al., 2014), and the POS embedding is randomly initialized for training with the whole GraphRel.

4.1.2 Bi-GCN

Since the original input sentence is a sequence and has no inherent graph structure to speak of, as Cetoli et al. (2016), we use dependency parser to create a dependency tree for the input sentence. We use the dependency tree as the input sentence's adjacency matrix and use GCN to extract regional dependency features.

The original GCN was designed for undirected graphs. To consider both incoming and outgoing word features, we follow Marcheggiani and Titov (2017) and implement bi-GCN as

$$\overrightarrow{h_u^{l+1}} = ReLU\left(\sum_{v \in \overrightarrow{N}(u)} \left(\overrightarrow{W}^l h_v^l + \overrightarrow{b}^l\right)\right)$$

$$\overleftarrow{h_u^{l+1}} = ReLU\left(\sum_{v \in \overleftarrow{N}(u)} \left(\overleftarrow{W}^l h_v^l + \overleftarrow{b}^l\right)\right)$$

$$h_u^{l+1} = \overrightarrow{h_u^{l+1}} \oplus \overleftarrow{h_u^{l+1}},$$

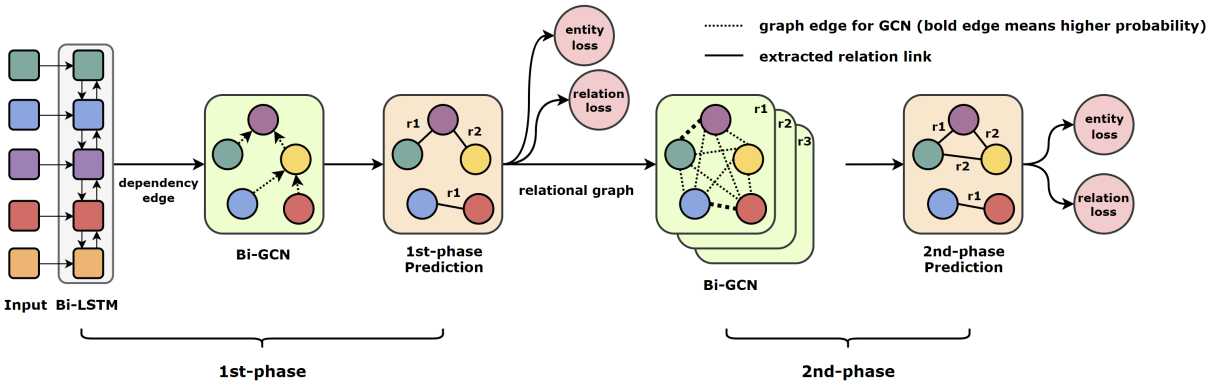

Figure 2: Overview of GraphRel with 2nd-phase relation-weighted GCN.

where h_u^l represents the hidden features of word u at layer l, $\overrightarrow{N}(u)$ contains all words outgoing from word u, and $\overleftarrow{N}(u)$ contains all words incoming to word u, both including word u itself. W and b are both learnable convolutional weights. $\overrightarrow{W}$, $\overrightarrow{b}$ and $\overleftarrow{W}$, $\overleftarrow{b}$ also represent the outgoing weight and incoming weight, respectively. We concatenate both outgoing and incoming word features as the final word feature.

4.1.3 Extraction of Entities and Relations

With the extracted word features from bi-RNN and bi-GCN, we predict the word entity and extract the relation for each word pair. For the word entity, we predict for all words according to the word features over 1-layer LSTM and apply categorical loss, denoted as $eloss_{1p}$, to train them.

For relation extraction, we remove the dependency edges and do prediction for all word pairs. For each relation r, we learn weight matrices W_r^1, W_r^2, W_r^3 and calculate the relation tendency score S as

$$S_{(w1,r,w2)} = W_r^3 ReLU\left(W_r^1 h_{w1} \oplus W_r^2 h_{w2}\right),$$

where $S_{(w1,r,w2)}$ represents the relation tendency score for $(w1, w2)$ under relation r and $(w1, w2)$ refers to the word pair. Note that $S_{(w1,r,w2)}$ should be different from $S_{(w2,r,w1)}$. For word pair $(w1, w2)$, we calculate all of the pair's relation tendency scores, including non-relation, and denote it as $S_{(w1,null,w2)}$. We apply the softmax function to $S_{(w1,r,w2)}$, yielding $P_r(w1, w2)$, which represents the probability of each relation r for $(w1, w2)$.

Since we extract relations for each word pair, our design includes no triplet count restrictions. By investigating the relations for each word pair,

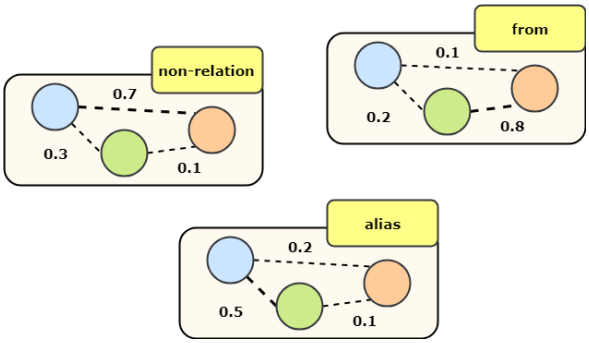

Figure 3: Relation-weighted graph for each relation.

GraphRel identifies as many relations as possible. With $P_r(w1, w2)$, we can also calculate the relation categorical loss here, denoted as $rloss_{1p}$. Please note that though both $eloss_{1p}$ and $rloss_{1p}$ will not be used as final prediction, they are also good auxiliary loss for training 1st-phase GraphRel.

4.2 2nd-phase Prediction

The extracted entities and relations in 1st-phase do not take each other into account. To consider interaction between named entities and relations, and to take account implicit features among all word pairs of the text, we present a novel 2nd-phase relation-weighted GCN for further extraction.

4.2.1 Relation-weighted Graph

After 1st-phase prediction, we build complete relation-weighted graph for each relation r where the edge of $(w1, w2)$ is $P_r(w1, w2)$, as shown in Fig. 3.

Then, 2nd-phase adopts bi-GCN on each relation graph which considers different influenced degrees of different relations and aggregates as the comprehensive word feature. The process can be

represented as

$$h_u^{l+1} = \textit{ReLU}\left(\sum_{v\in V}\sum_{r\in R} P_r(u,v) \times \left(W_r^l h_v^l + b_r^l\right)\right) + h_u^l,$$

where $P_r(u,v)$ represents the edge weight (the probability of word u to word v under relation r). W_r and b_r means the GCN weight under relation r. V includes all words and R contains all relations. Note that the complete bi-GCN also takes both the incoming and outgoing situations into account. The bi-GCN in 2nd-phase further considers the relation-weighted propagations and extracts more sufficient features for each word.

With the newer word features from 2nd-phase, we perform named entity and relation classification again for more robust relation prediction. The losses for these are denoted as $\textit{eloss}_{2p}$ and $\textit{rloss}_{2p}$.

4.3 Training Detail

We use two kinds of loss in GraphRel: entity loss and relation loss, both of which belong to categorical loss. For entity loss, we use the conventional (*Begin*, *Inside*, *End*, *Single*, *Out*) tagging for the ground-truth labels. Each word belongs to one of the five classes. The ground-truth entity label for $\textit{eloss}_{1p}$ and $\textit{eloss}_{2p}$ are the same; we use cross-entropy as the categorical loss function during training.

For relation loss, we feed in a one-hot relation vector as the ground truth of $P_r(w1, w2)$ for each word pair $(w1, w2)$. Since we predict relations based on word pairs, the ground truth should likewise be based on word pairs. That is, word *United* has a *HasPresident* relation to both word *Barack* and word *Obama*, as does word *States*. We believe that this word-pair-based relation representation provides GraphRel with the information it needs to learn to extract relations. The ground-truth relation vector for $\textit{rloss}_{1p}$ and $\textit{rloss}_{2p}$ are the same. As entity loss, we also use cross-entropy as the categorical loss function during training.

For both *eloss* and *rloss*, we add an additional double-weight for those in-class entity or relation terms. Finally, the total loss is calculated as the sum of all entity loss and relation loss:

$$loss_{all} = \left(\textit{eloss}_{1p} + \textit{rloss}_{1p}\right) + \alpha\left(\textit{eloss}_{2p} + \textit{rloss}_{2p}\right),$$

where α is a weight between loss of 1st-phase and 2nd-phase. We minimize $loss_{all}$ and train the whole GraphRel in an end-to-end manner.

4.4 Inference

During inference, the baseline prediction method is **head** prediction, where a relation triplet such as (*BarackObama*, *PresidentOf*, *UnitedStates*) is extracted if and only if *BarackObama*, *UnitedStates* are both identified as entity mentions and *PresidentOf* is the most probable class of $P_{(Obama,States)}$.

Another baseline extraction method that might be more stable is **average** prediction, where all word pairs between an entity mention pair are taken into account and decide a relation with maximum averaged probability.

Finally, we propose a **threshold** prediction method, where all word pairs of an entity mention pair are still taken into account but in an independent fashion. For example, if 2 of the 4 distributions have *PresidentOf* as the most probable class, then the triplet (*BarackObama*, *PresidentOf*, *UnitedStates*) is extracted only if $2/4 = 50\% > \theta$ where θ is a free threshold parameter. This way, users can select their preferred precision and recall trade-off by adjusting θ. In the experiments, if not specified, threshold inference with $\theta = 0$ is used.

5 Experiments

In this section, we present the experimental results of the proposed GraphRel. We first describe implementation details, the datasets, and the baselines we compare with. Then we show the quantitative results for two datasets, conduct detailed analyses, and different categories of named entities. Finally, we demonstrate the improved effect of 2nd-phase via a case study.

5.1 Experimental Settings

In our implementation, we chose the pre-trained GloVe (300d) as a fixed word embedding. The word embedding was then concatenated with a trainable POS embedding (15d) as the final input embedding for each word. The POS tag for each word and the dependency tree for whole sentences was retrieved from spaCy (Honnibal and Johnson, 2015).

We use bi-LSTM with 256 units and 2-layer bi-GCN with 256 feature size in 1st-phase. For the 2nd-phase, the relation-weighted bi-GCN is 1-layer, also with a feature size of 256. During training, we set the LSTM dropout rate to 0.5, the learning rate to 0.0008, and the loss weight α to 3. We train GraphRel using the Adam (Kingma

Category	NYT		WebNLG	
	Train	Test	Train	Test
Normal	37013	3266	1596	246
EPO	9782	978	227	26
SEO	14735	1297	3406	457
All	56195	5000	5019	703
#Relation	24		246	

Table 1: Statistics of dataset.

Method	NYT			WebNLG		
	Precision	Recall	F1	Precision	Recall	F1
NovelTagging	62.4%	31.7%	42.0%	**52.5%**	19.3%	28.3%
OneDecoder	59.4%	53.1%	56.0%	32.2%	28.9%	30.5%
MultiDecoder	61.0%	56.6%	58.7%	37.7%	36.4%	37.1%
GraphRel$_{1p}$	62.9%	57.3%	60.0%	42.3%	39.2%	40.7%
GraphRel$_{2p}$	**63.9%**	**60.0%**	**61.9%**	44.7%	**41.1%**	**42.9%**

Table 2: Results for both NYT and WebNLG datasets.

and Ba, 2015) optimizer and implement it under PyTorch.

5.1.1 Dataset

We use the NYT (Riedel et al., 2010) and WebNLG (Gardent et al., 2017) datasets to evaluate the proposed method. As NovelTagging and MultiDecoder, for NYT, we filter sentences with more than 100 words and for WebNLG, we use only the first sentence in each instance in our experiments. The statistics of NYT and WebNLG is described in Table. 1.

We divided relation triplets into three categories: *Normal*, *EntityPairOverlap (EPO)*, and *SingleEntityOverlap (SEO)*. The counts for each category are also shown in Table 1. Since one entity belonged to several different relations, *EntityPairOverlap* and *SingleEntityOverlap* were more difficult tasks. We discuss the result for different categories in the detailed analysis.

5.2 Baseline and Evaluation Metrics

We compared GraphRel with two baselines: NovelTagging (Zheng et al., 2017) and MultiDecoder (Zeng et al., 2018). NovelTagging is a sequence tagger which predicts both entity and relation classes for each sentence word. MultiDecoder is a state-of-the-art method that considers relation extraction as a seq-seq problem and uses dynamic decoders to extract relation triplets. The results for both baselines come directly from the original papers.

As two baselines, we adopted the standard F1 score to evaluate the results. The predicted triplets were seen as correct if and only if the relation and the head of the two corresponding entities were the same as the ground truth.

5.3 Quantitative Results

Table 2 presents the precision, recall, and F1 score of NovelTagging, MultiDecoder, and GraphRel for both the NYT and WebNLG datasets. OneDecoder, proposed in MultiDecoder's original paper, uses only a single decoder to extract relation triplets. GraphRel$_{1p}$ is the proposed method but only 1st-phase, and GraphRel$_{2p}$ is the complete version, which predicts relations and entities after the 2nd-phase.

For the NYT dataset, we see that GraphRel$_{1\text{-hop}}$ outperforms NovelTagging by 18.0%, OneDecoder by 4.0%, and MultiDecoder by 1.3% in terms of F1. As it acquires both sequential and regional dependency word features, GraphRel$_{1\text{-hop}}$ performs better on both precision and recall, resulting in a higher F1 score. With relation-weighted GCN in 2nd-phase, GraphRel$_{2p}$, which considers interaction between name entities and relations, further surpasses MultiDecoder by 3.2% and yields a 1.9% improvement in comparison with GraphRel$_{1p}$.

Similar results can be found on the WebNLG dataset: GraphRel$_{1p}$ outperforms baseline F1 scores by 3.6%, and GraphRel$_{2p}$ further improves 2.2% upon GraphRel$_{1p}$. From the NYT and WebNLG results, we show that GCN's regional dependency feature and 2nd-phase prediction both aid relation prediction in terms of precision, recall, and F1 score.

NovelTagging and MultiDecoder both use a sequential architecture. As NovelTagging assumes that an entity belongs to a single relation, precision is high but recall is low. MultiDecoder uses a dynamic decoder to generate relation triplets. Because of the innate restrictions on RNN unrolling, the number of triplets it can generate is limited. However, for GraphRel, as we predict relations for each word pair, we are free of that restriction. We believe that GraphRel is the most balanced method as it maintains both high precision and high recall, yielding higher F1 scores.

5.4 Detailed Analysis

To further analyze the proposed GraphRel, we present the results under different types of triplets, the improvement over name entity recognition, different numbers of GCN layer used, and different inference methods.

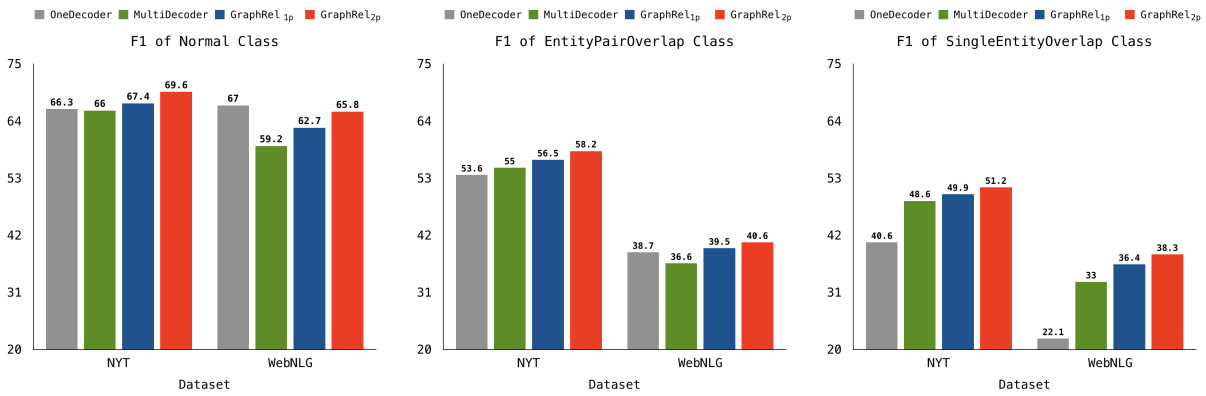

Figure 4: Results (F1 score) by named entity category.

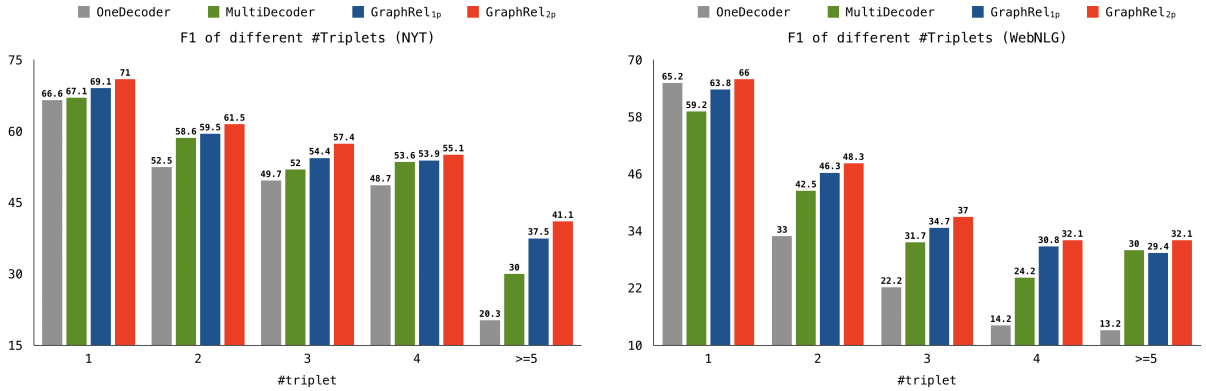

Figure 5: Results (F1 score) by sentence triplet count.

Method	NYT	WebNLG
GraphRel$_{1p}$	88.8%	89.1%
GraphRel$_{2p}$	**89.2%**	**91.9%**

Table 3: Results (F1 score) of entity recognition for GraphRel.

5.4.1 Different Types of Triplets

We first investigate the results under different entity categories. Fig. 4 presents the results for both NYT and WebNLG datasets.

For GraphRel, as we predict relations for all word pairs, all words can have relations with other words: thus entity overlap is not a problem. Though MultiDecoder tries to use a dynamic decoder, the result shows that GraphRel surpasses them in all entity categories. For instance, on the WebNLG dataset, GraphRel$_{1p}$ outperforms MultiDecoder by 3.5% on the normal class, 2.9% on the EPO class, and 3.4% on the SEO class. And GraphRel$_{2p}$ further improves GraphRel$_{1p}$ for each class.

We also compare the results given different numbers of triplets in a sentence, as illustrated as Fig. 5. The x-axis represents 1, 2, 3, 4, or more than 5 triplets in a sentence. Because of the single decoder, OneDecoder performs well for single triplets, but performance drops drastically for more triplets in a sentence. As with the experiment for different entity categories, GraphRel$_{1p}$ and GraphRel$_{2p}$ both outperform the baselines under all numbers of triplets in a sentence. GraphRel$_{1p}$ outperforms MultiDecoder by 7.5% for more than 5 triplets in a sentence and GraphRel$_{2p}$ further surpasses MultiDecoder by 11.1% on NYT.

5.4.2 Improvement over Entity Recognition and Different Numbers of GCN Layer

From Table. 3, GraphRel$_{2p}$ can surpass 1st-phase by 0.4% and 2.8% for entity recognition on both NYT and WebNLG. It also shows that 2nd-phase relation-weighted GCN is effective on not only relation extraction but also name entity recognition.

To confirm that our 2-layer 1st-phase added 1-layer 2nd-phase is the best setting, we investigate the result of different numbers of GCN layer used in both 1st-phase and 2nd-phase. Table. 5 presents the results of using 3 GCN layers for 1st-phase and

Sentence	GraphRel$_{1p}$	GrapRel$_{2p}$
Agra Airport is in India where one of its leaders is Thakur.	(Agra Airport, location, India) (India, leader_name, Thakur)	(Agra Airport, location, India) (India, leader_name, Thakur)
In Italy, the capital is Rome and A.S. Gubbio 1910 is located there.	(Italy, captical, Rome)	(Italy, captical, Rome) (A.S. Gubbio 1910, ground, Italy)
Asam pedas (aka Asam padeh) is from the Sumatra and Malay Peninsula regions of Malaysia.	(Asam pedas, alias, Asam padeh) (Asam pedas, region, Malay Peninsula) (Asam pedas, country, Malaysia)	(Asam pedas, alias, Asam padeh) (Asam pedas, region, Malay Peninsula) (Asam padeh, region, Malay Peninsula) (Asam pedas, country, Malaysia) (Asam padeh, country, Malaysia)

Table 4: Case Study for Graph$_{1p}$ and GraphRel$_{2p}$.

Phase	#GCN layer	NYT	WebNLG
1st-phase	2	60.0%	**40.7%**
	3	60.0%	40.5%
2nd-phase†	1	**61.9%**	**42.9%**
	2	61.6%	42.4%
3rd-phase‡	1	61.8%	42.7%

Table 5: Results (F1 score) by different numbers of GCN layer.

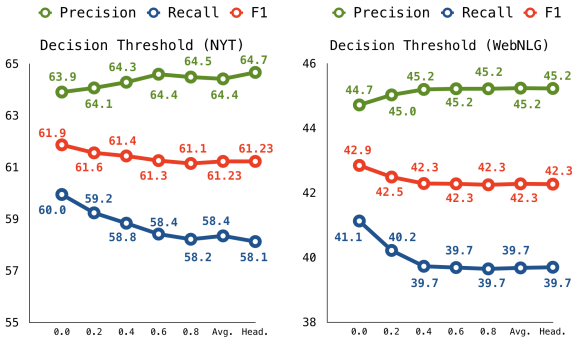

Figure 6: Results by different decision thresholds.

2 layers of relation-weighted GCN for 2nd-phase. However, it shows that more GCN layers can not bring out better prediction and our (2, 1) layer setting should be the most suitable one for relation extraction task.

We also experiment on 3rd-phase method, adopting relation-weighted GCN again where the graph is based on 2nd-phase's predicited relations. And it shows that our 2nd-phase is sufficient enough for relation extraction.

5.4.3 Inference Methods

We compare the two baseline inference methods, head and average, and the threshold method under different θ. Fig. 6 shows their results when applied to GraphRel$_{2p}$ on NYT and WebNLG.

It can be seen that the threshold inference method efficaciously adjusts the trade-off between precision and recall with different choices of θ. By reducing the threshold from $\theta = 0.8$ to $\theta = 0$, the recall is significantly increased by 1.8% and 1.4% respectively on NYT and WebNLG, with only a marginal 0.6% loss of precision. The effectiveness of the proposed threshold method then leads to the best performance on both datasets, surpassing both the head and average ones.

†#GCN layer in 1st-phase set to 2.

‡#GCN layer in 1st-phase and 2nd-phase set to 2 and 1.

5.5 Case Study

Table. 4 shows the case study of our proposed GraphRel. The first sentence is an easy case and both GraphRel$_{1p}$ and GraphRel$_{2p}$ can extract accurately. For the second case, although *there* does not belong to name entity, it should contain the hidden semantic of *Italy*. Therefore, 2nd-phase can further predict that *A.S. Gubbio 1910* grounds in *Italy*. The third case is an *SEO* class in which GraphRel$_{1p}$ discovers that *Asam pedas* is the same as Asam padeh, thus the latter should also locate in *Malay Peninsula* and come from *Malaysia*.

6 Conclusion

In this paper, we present **GraphRel**, an end-to-end relation extraction model which jointly learns named entities and relations based on graph convolutional networks (GCN). We combine RNN and GCN to extract not only sequential features but also regional dependency features for each word. Implicit features among all word pairs of the text are also considered in our approach. We predict relations for each word pair, solving the problem of entity overlapping. Furthermore, we introduce a novel relation-weighted GCN that considers interactions between named entities and relations. We evaluate the proposed method on the NYT and WebNLG datasets. The results show that our method outperforms previous work by 3.2% and 5.8% and achieves a new state-of-the-art for relation extraction.

References

A. Cetoli, S. Bragaglia, A.D. O'Harney, and M. Sloan. 2016. Graph convolutional networks for named entity recognition. In *Proceedings of TLT*.

Yee-Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of ACL*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planners. In *Proceedings of ACL*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Thomas Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of EMNLP*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of ACL*.

Bang Liu, Ting Zhang, Di Niu, Jinghong Lin, Kunfeng Lai, and Yu Xu. 2018. Matching long text documents via graph convolutional networks. *arXiv preprint arXiv:1802.07459*.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of NAACL*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*.

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

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of EMNLP*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Yujie Qian, Enrico Santus, Zhijing Jin, Jiang Guo, and Regina Barzilay. 2019. Graphie: A graph-based framework for information extraction. In *Proceedings of NAACL*.

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 *Proceedings of WWW*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP-CoNLL*.

Xu Yan, 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 EMNLP*.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of COLING*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research (JMLR)*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*.

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 ACL*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*.

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 *Proceedings of ACL*.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL*.