Joint Type Inference on Entities and Relations via Graph Convolutional Networks

Changzhi Sun [1,*], **Yeyun Gong**[2], **Yuanbin Wu**[1,3], **Ming Gong**[2], **Daxing Jiang**[2], **Man Lan**[1], **Shiliang Sun**[1], **and Nan Duan**[2]

[1]Department of Computer Science and Technology, East China Normal University
[2]Microsoft Research Asia
[3]State Key Laboratory of Cognitive Intelligence, iFLYTEK
{changzhisun}@stu.ecnu.edu.cn {ybwu,mlan,slsun}@cs.ecnu.edu.cn
{yegong, nanduan, migon, djiang}@microsoft.com

Abstract

We develop a new paradigm for the task of joint entity relation extraction. It first identifies entity spans, then performs a joint inference on entity types and relation types. To tackle the joint type inference task, we propose a novel graph convolutional network (GCN) running on an entity-relation bipartite graph. By introducing a binary relation classification task, we are able to utilize the structure of entity-relation bipartite graph in a more efficient and interpretable way. Experiments on ACE05 show that our model outperforms existing joint models in entity performance and is competitive with the state-of-the-art in relation performance.

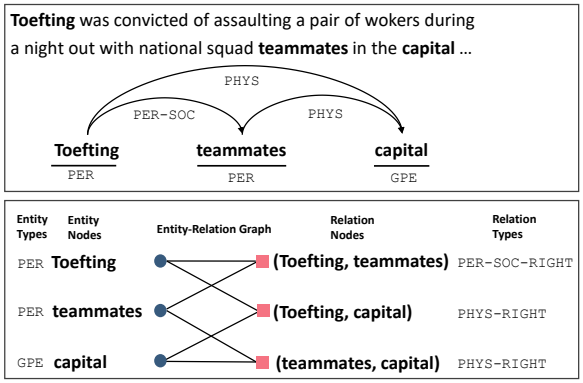

Figure 1: An example from ACE05. The first part contains annotations and the second part is the entity-relation graph of the sentence used in GCN.

1 Introduction

Extracting entities and relations from plain texts is an important and challenging task in natural language processing. Given a sentence, the task aims to detect text spans with specific types (*entities*) and semantic relations among those text spans (*relations*). For example, in the Figure 1, "Toefting" is a person entity (PER), "teammates" is a person entity (PER), and the two entities have a Person-Social relation (PER-SOC).

To tackle the task of entity relation extraction, various methods have been proposed, which can be divided into two categories: pipeline models and joint models. Pipeline models extract entities and relations in two stages: entities are first extracted by an entity model, and then these extracted entities are used as the inputs of a relation model. Pipeline models often ignore interactions between the two models and they suffer from error propagation. Joint models integrate information between entities and relations into a single model with the joint training, and have achieved better results than the pipeline models. In this paper, we focus on joint models.

More and more joint methods have been applied to this task. Among them, Miwa and Bansal (2016); Katiyar and Cardie (2017) identify the entity with a sequence labelling model, and identify the relation type with a multi-class classifier. These joint methods do joint learning through sharing parameters and they have no explicit interaction in type inference. In addition, some complex joint decoding algorithms (e.g., simultaneously decoding entities and relations in beam search) have been carefully investigated, including Li and Ji (2014); Zhang et al. (2017); Zheng et al. (2017); Wang et al. (2018). They jointly handle span detection and type inference to achieve more interactions.

By inspecting the performance of existing models (Sun et al., 2018) on ACE05, we find that, for many entities, their spans are correctly identified, but their entity types are wrong. In particular, the F1 of extracting typed entities is about 83%, while the F1 of extracting entity spans is about 90%. Thus, if we have a better type inference model, we

[*] Work done while this author was an intern at Microsoft Research Asia.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1361–1370
Florence, Italy, July 28 - August 2, 2019.

may get a better joint extraction performance. At the same time, we observe that a joint inference on entity and relation types could be potentially better than predicting them independently. For example, in Figure 1, the PER−SOC relation suggests that the type of "Toefting" might be PER, and vice versa. Moreover the PER ("Toefting") and the relation PER−SOC could benefit from other relations such as PHYS.

In this paper, we define joint entity relation extraction into two sub-tasks: entity span detection and entity relation type deduction. For entity span detection, we treat it as a sequence labeling problem. For joint type inference, we propose a novel and concise joint model based on graph convolutional networks (GCNs) (Kipf and Welling, 2017). The two sub-models are trained jointly. Specifically, given all detected entity spans in a sentence, we define an entity-relation bipartite graph. For each entity span, we assign an entity node. For each entity-entity pair, we assign a relation node. Edges connect relation nodes and their entity nodes (last part of Figure 1). With efficient graph convolution operations, we can learn representations for entity nodes and relation nodes by recursively aggregating information from their neighborhood over the bipartite graph. It helps us to concisely capture information among entities and relations. For example, in Figure 1, to predict the PER ("Toefting"), our joint model can pool the information of PER−SOC, PHYS, PER ("teammates") and GPE (captital).

To further utilize the structure of the graph, we also propose assigning different weights on graph edges. In particular, we introduce a binary relation classification task, which is to determine whether the two entities form a valid relation. Different from previous GCN-based models (Shang et al., 2018; Zhang et al., 2018), the adjacency matrix of graph is based on the output of binary relation classification, which makes the proposed adjacency matrix more explanatory. To summarize, the main contributions of this work are [1]

- We present a novel and concise joint model to handle the joint type inference problem based on graph convolutional network (GCN).
- We introduce a binary relation classification task to explore the structure of entity-relation bipartite graph in a more efficient and interpretable way.
- We show that the proposed joint model on ACE05 achieves best entity performance, and is competitive with the state-of-the-art in relation performance.

2 Background of GCN

In this section, we briefly describe graph convolutional networks (GCNs). Given a graph with n nodes, the goal of GCNs is to learn structure-aware node representations on the graph which takes as inputs:

- an $n \times d$ input node embedding matrix $\mathbf{H}$, where n is the number of nodes and d is the dimension of input node embedding;
- an $n \times n$ matrix representation of the graph structure such as the adjacency matrix $\mathbf{A}$ (or some function thereof) [2].

In an L-layer GCNs, every layer can be written as a non-linear function

$$\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \quad (1)$$

with $\mathbf{H}^{(0)} = \mathbf{H}$, where $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix and $\mathbf{W}^{(l)}$ is a parameter matrix for the l-th GCN layer. $\mathbf{D}$ is the diagonal node degree matrix, where $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. σ is a non-linear activation function like ReLU. Finally, we can obtain a node-level output $\mathbf{Z} = \mathbf{H}^{(L)}$, which is an $n \times d$ feature matrix.

3 Approach

We define the joint entity relation extraction task. Given a sentence $s = w_1, \ldots w_{|s|}$ (w_i is a word), the task is to extract a set of entity spans $\mathcal{E}$ with specific types and a set of relations $\mathcal{R}$. An entity span $e \in \mathcal{E}$ is a sequence of words labeling with an entity type y (e.g., person (PER), organization (ORG)). A relation r is a quintet (e_1, y_1, e_2, y_2, l), where e_1 and e_2 are two entity spans with specific types y_1 and y_2. l is a relation type describing the semantic relation between two entities. (e.g., organization affiliation relation (ORG−AFF)). Let $\mathcal{T}_e$, $\mathcal{T}_r$ be the set of possible entity types and relation types respectively.

[1] Our implementation is available at https://github.com/changzhisun/AntNRE.

[2] In order to incorporate self-information, we add a self-loop to each node, where $\mathbf{A}_{ii} = 1.0$ for each node i.

In this work, we decompose the joint entity relation extraction task into two parts, namely, entity span detection and entity relation type deduction. We first treat entity span detection as a sequence labelling task (Section 3.1), and then construct an entity-relation bipartite graph (Section 3.2) to perform joint type inference on entity nodes and relation nodes (Section 3.3). All submodels share parameters and are trained jointly. Different from existing joint learning algorithms (Sun et al., 2018; Zhang et al., 2017; Katiyar and Cardie, 2017; Miwa and Bansal, 2016), we propose a concise joint model to perform joint type inference on entities and relations based on GCNs. It considers interactions among multiple entity types and relation types simultaneously in a sentence.

3.1 Entity Span Detection

To extract entity spans from a sentence (Figure 2), we adopt the BILOU sequence tagging scheme: B, I, L and O denote the begin, inside, last and outside of a target span, U denotes a single word span. For example, for a person (PER) entity "Patrick McDowell", we assign B to "Patrick" and L to "McDowell".

Given an input sentence s, we use a bidirectional long short term memory (biLSTM) network (Hochreiter and Schmidhuber, 1997) with parameter θ_{seq} to incorporate information from both forward and backward directions of s.

$$\mathbf{h}_i = \text{biLSTM}(\mathbf{x}_i; \theta_{\text{seq}}), \quad (2)$$

where $\mathbf{h}_i$ is the concatenation of a forward and a backward LSTM's hidden states at position i, and $\mathbf{x}_i$ is the word representation of w_i which contains pre-trained embeddings and character-based word representations generated by running a CNN on the character sequences of w_i. Then, we employ a softmax output layer to predict w_i's tag $\hat{t}_i$,

$$P(\hat{t}_i|s) = \text{Softmax}(\mathbf{W}_{\text{span}}\mathbf{h}_i),$$

where $\mathbf{W}_{\text{span}}$ is the parameter. Given an input sentence s and its gold tag sequence $\mathbf{t} = t_1, \ldots, t_{|s|}$, the training objective is to minimize [3]

$$\mathcal{L}_{\text{span}} = -\frac{1}{|s|}\sum_{i=1}^{|s|} \log P(\hat{t}_i = t_i|s). \quad (3)$$

[3] We have also tried biLSTM-CRF (Huang et al., 2015) as an advanced sequence labelling model, but performances are nearly the same in our experiments.

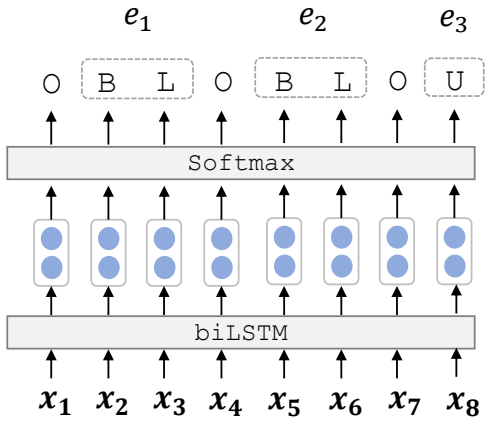

Figure 2: The biLSTM model for entity span detection.

3.2 Entity-Relation Bipartite Graph

Given a set of detected entity spans $\hat{\mathcal{E}}$ (obtained from the entity span tag sequence $\hat{\mathbf{t}}$), we consider all entity span pairs in $\hat{\mathcal{E}}$ as candidate relations [4]. Then we build a heterogeneous undirected bipartite graph $\mathcal{G}^s$ which contains entity nodes and relation nodes in a sentence s. In the graph $\mathcal{G}^s$, interactions on multiple entity types and relation types can be explicitly modeled. The number of nodes n in the graph $\mathcal{G}^s$ is the number of entity spans $|\hat{\mathcal{E}}|$ plus the number of all candidate relations $\frac{|\hat{\mathcal{E}}|(|\hat{\mathcal{E}}|-1)}{2}$. We have an initial input node embedding matrix $\mathbf{H}$. For a relation r_{12} and its two entities e_1, e_2, we use $\mathbf{H}_{r_{12}}$ to denote relation embedding of r_{12}, and use $\mathbf{H}_{e_1}$,$\mathbf{H}_{e_2}$ to denote entity embedding of e_1, e_2 respectively.

Next, we build edges between entity nodes and relation nodes. For graph edges, we connect every relation node to its two entity nodes instead of directly connecting any entity (relation) nodes. Thus we focus on the bipartite graph. The reasons are two folds. a) We do not think that all the remaining entities in the sentence are helpful. Relation nodes are bridges between entity nodes and vice versa. b) GCN is not suitable for fully-connected graphs because GCN reduce to rather trivial operations on fully-connected graphs. It means that, for an entity node e, the only way to observe other entities is through relations which e takes part in. For example, given a relation node r_{12} and its two entity nodes e_1, e_2, we add two edges. One is the edge between e_1 and r_{12}, and another is the edge

[4] The first entity span is always on the left side of the second entity span of each candidate relation, and we use in total $2\mathcal{T}_r + 1$ relation types in order to consider both directions. The additional type is the None which means no relation between entity span pair.

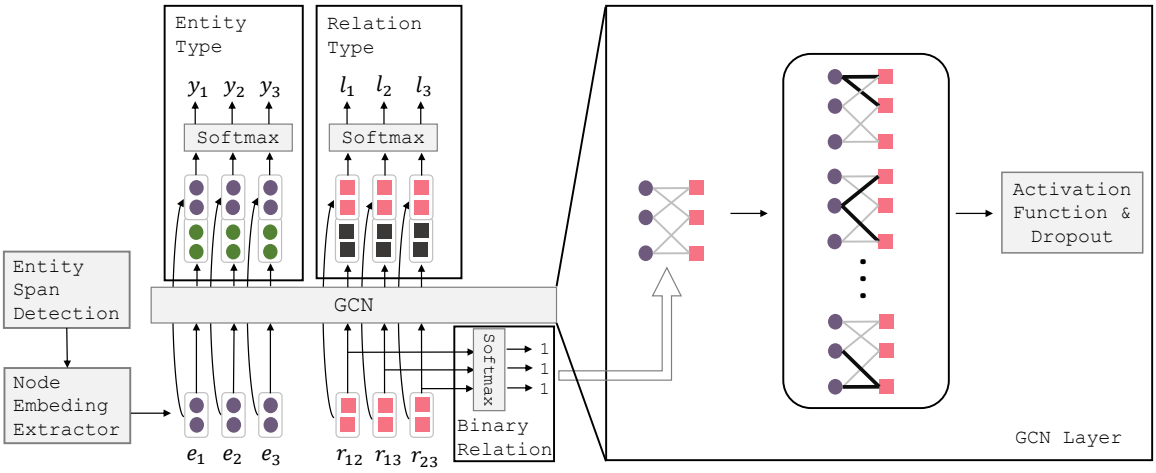

Figure 3: Our network structure for the joint entity and relation extraction based on GCN. The node embedding extractor computes $\mathbf{H}_e$ and $\mathbf{H}_r$.

between e_2 and r_{12}. We refer to it as **static** graph.

In order to further utilize the structure of the graph (some kind of prior knowledge) instead of using a static graph, we also investigate the **dynamic** graph for pruning redundant edges. A key intuition is that if two entities hold a relation, we could add two edges between the relation node and two entity nodes. Conversely, if two entities have no relation, we keep two entity nodes and the relation node separately. To this end, we introduce the **binary relation classification** task. It aims to predict whether a certain relation exists between an entity span pair (ignoring specific relation types). We build a binary relation model which predicts a label in $\{0, 1\}$ to indicate the existence of a candidate relation based on relation node embedding. Given a relation node r_{ij} in a sentence s, to get the posterior of the binary relation label $\hat{b}$, we apply softmax layer on the relation node embedding $\mathbf{H}_{r_{ij}}$,

$$P(\hat{b}|r_{ij}, s) = \text{Softmax}(\mathbf{W}_{\text{bin}}\mathbf{H}_{r_{ij}}),$$

where $\mathbf{W}_{\text{bin}}$ is the parameter. The training objective is to minimize

$$\mathcal{L}_{\text{bin}} = -\sum_{r_{ij}} \frac{\log P(\hat{b} = b|r_{ij}, s)}{\#\text{ candidate relations } r_{ij}}, \quad (4)$$

where true binary annotations b are transformed from the original typed relation labels. Formally, the adjacency matrix $\mathbf{A}$ is defined as

- if $P(\hat{b} = 1|r_{ij}, s) > 0.5$, we set the value of $\mathbf{A}$ between entity nodes e_i, e_j and relation node r_{ij} to 1.0,
- the diagonal elements of $\mathbf{A}$ are set to 1.0,
- while others are set to 0.0.

To compare with **hard** binary value $\mathbf{A}$, we also try the **soft** value $\mathbf{A}$ in experiments. It means that we set the value of $\mathbf{A}$ between entity nodes e_i, e_j and relation node r_{ij} to the probability $P(\hat{b} = 1|r_{ij}, s)$ except for the diagonal elements (they are set to 1.0).

Here, we introduce how to compute two types of contextualized node embedding in the graph $\mathcal{G}^s$: entity node embedding and relation node embedding.

Entity Node Embedding Given an entity span $e \in \hat{\mathcal{E}}$, for each word $w_i \in e$, we first collect w_i's biLSTM hidden vector $\mathbf{h}_i$ from entity span model. Then, we use a CNN (a single convolution layer with a max-pooling layer) with a multi-layer perceptron (MLP) on vectors $\{\mathbf{h}_i|w_i \in e\}$ to obtain the resulting d-dimensional entity span node embedding $\mathbf{H}_e$ ($\mathbf{H}$ is a matrix mentioned before in Section 2), as shown in the left part of Figure 4.

Relation Node Embedding Given a candidate relation r_{12}, we extract two types of features, namely, features regarding words in e_1, e_2 and features regarding contexts of the entity span pair (e_1, e_2). For features on words in e_1, e_2, we simply use entity node embedding $\mathbf{H}_{e_1}$ and $\mathbf{H}_{e_2}$. For context features of the entity span pair (e_1, e_2), we build three feature vectors by looking at words between e_1 and e_2, words on the left of the pair and words on the right of the pair. Similarly, we build three features by running another CNN with

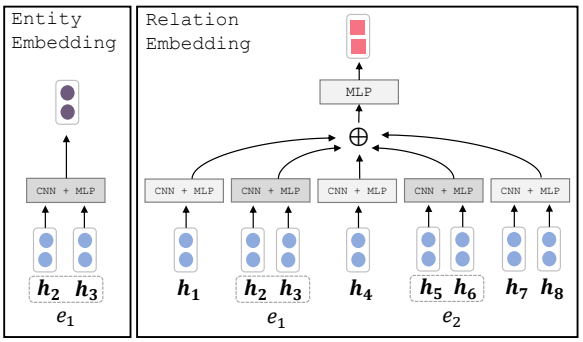

Figure 4: Our node embedding extractor.

an MLP. Finally, the five feature vectors are concatenated to a single vector. To get d-dimensional relation node embedding $\mathbf{H}_{r_{12}}$, we apply an MLP on the single vector, as shown in the right part of Figure 4.

3.3 Joint Type Inference

After building the entity-relation bipartite graph, we feed the graph into a multi-layer GCNs to obtain the node-level output $\mathbf{Z}$. For each row in $\mathbf{Z}$ (entity or relation node representation), it can gather and summarize information from other nodes in the graph $\mathcal{G}^s$ although there is no direct entity-entity or relation-relation edges in the graph. Then the final node representation $\mathbf{F}$ of graph $\mathcal{G}^s$ is concatenated by the input node embedding $\mathbf{H}$ and the node-level output $\mathbf{Z}$ ($\mathbf{H}, \mathbf{Z}$ and $\mathbf{F}$ are matrices).

Given an entity node e_i and a relation node r_{ij}, to predict the corresponding node types, we pass the resulted node representation into two fully connected layer with a softmax function, respectively,

$$P(\hat{y}|e_i, s) = \text{Softmax}(\mathbf{W}_{\text{ent}}\mathbf{F}_{e_i}),$$

$$P(\hat{l}|r_{ij}, s) = \text{Softmax}(\mathbf{W}_{\text{rel}}\mathbf{F}_{r_{ij}}),$$

where $\mathbf{W}_{\text{ent}}$, $\mathbf{W}_{\text{rel}}$ are parameters. And the training objective is to minimize

$$\mathcal{L}_{\text{ent}} = -\frac{1}{|\hat{\mathcal{E}}|}\sum_{e_i \in \hat{\mathcal{E}}} \log P(\hat{y} = y|e_i, s), \quad (5)$$

$$\mathcal{L}_{\text{rel}} = -\sum_{r_{ij}} \frac{\log P(\hat{l} = l|r_{ij}, s)}{\#\text{ candidate relations } r_{ij}}, \quad (6)$$

where the true label y, l can be read from annotations, as shown in Figure 3.

3.4 Training

To train the joint model, we optimize the combined objective function $\mathcal{L} = \mathcal{L}_{\text{span}} + \mathcal{L}_{\text{bin}} + \mathcal{L}_{\text{ent}} + \mathcal{L}_{\text{rel}}$, where the training is accomplished by the shared parameters. We employ the scheduled sampling strategy (Bengio et al., 2015) in the entity model similar to (Miwa and Bansal, 2016). We optimize our model using Adadelta (Zeiler, 2012) with gradient clipping. The network is regularized with dropout. Within a fixed number of epochs, we select the model according to the best relation performance on development sets[5].

4 Experiments

We conduct experiments on **ACE05** dataset, which is a standard corpus for the entity relation extraction task. It includes 7 entity types and 6 relation types between entities. We use the same data split of ACE05 documents as previous work (351 training, 80 development and 80 testing) (Miwa and Bansal, 2016).

We evaluate the performances using precision (P), recall (R) and F1 scores following (Miwa and Bansal, 2016; Sun et al., 2018). Specifically, an output entity (e, y) is correct if its type y and the region of its head e are correct, and an output relation r is correct if its (e_1, y_1, e_2, y_2, l) are correct (i.e., exact match).

In this paper, the default setting "GCN" is the 1-layer GCN-based joint model with the dynamic hard adjacency matrix, which achieves the best relation performance on ACE05 dataset.

4.1 End-to-End Results on ACE05

First, we compare proposed models with previous work in Table 1. In general, our "GCN" achieves the best entity performance 84.2 percent comparing with existing joint models. For relation performance, our "GCN" significantly outperforms all joint models except for (Sun et al., 2018) which uses more complex joint decoder. Comparing with our basic neural network "NN", our "GCN" has large improvement both on entities and relations. Those observations demonstrate the effectiveness of our "GCN" for capturing information on multiple entity types and relation types from a sentence.

[5] Our word embeddings is initialized with 100-dimensional glove (Pennington et al., 2014) word embeddings. The dimensionality of the hidden units and node embedding are set to 128. For all CNN in our network, the kernel sizes are 2 and 3, and the output channels are 25.

Model	Entity			Relation		
	P	R	F	P	R	F
L&J (2014)	85.2	76.9	80.8	65.4	39.8	49.5
Zhang (2017)	-	-	83.5	-	-	57.5
Sun (2018)	83.9	83.2	83.6	64.9	**55.1**	**59.6**
M&B (2016)	82.9	**83.9**	83.4	57.2	54.0	55.6
K&C (2017)	84.0	81.3	82.6	55.5	51.8	53.6
NN	85.7	82.1	83.9	65.6	50.7	57.2
GCN	**86.1**	82.4	**84.2**	**68.1**	52.3	59.1

Table 1: Results on the ACE05 test data. Li and Ji (2014) Zhang et al. (2017) and Sun et al. (2018) are joint decoding algorithms. Miwa and Bansal (2016) and Katiyar and Cardie (2017) are joint training systems without joint decoding. "NN" is our neural network model without GCN. "GCN" is dynamic hard GCN-based neural network. We omit pipeline methods which underperform joint models (see (Li and Ji, 2014) for details).

	1-layer	2-layer	3-layer
F1 of Entity Span	90.4	90.5	**90.7**
F1 of Binary Relation	61.5	**62.9**	62.8
F1 of Entity	81.6	82.1	**82.2**
F1 of Relation	**53.8**	53.5	53.6

Table 2: Results on the ACE05 development set with respect to the number of GCN layers.

Compared to the state-of-the-art method which adopts minimum risk training (Sun et al., 2018), our "GCN" has better entity performance and comparable relation performance. Different from existing joint decoding systems, we do not use complex joint decoding algorithms such as beam search (Li and Ji, 2014), global normalization (Zhang et al., 2017) and minimum risk training (Sun et al., 2018). Our models only rely on sharing parameters similar to (Miwa and Bansal, 2016; Katiyar and Cardie, 2017). It is worth noting that the precision of our "GCN" is high compared to all the other methods. We attribute the phenomenon to the strong ability to model feature representations of entity nodes and relation nodes.

Next, we evaluate our model with different settings. As mentioned in Section 3.2, we have three types of graph: "GCN (static)", "GCN (dynamic + hard)" and "GCN (dynamic + soft)". The last three rows of Table 3 show their performances. We have three observations regarding the Table 3.

1. Compared with "Sun (NN)" model which is the base neural network without minimum risk training (Sun et al., 2018), our "NN" performs better 0.5 point on entities. One reason might be the entity type model and the relation type model share more parameters (entity CNN+MLP parameters), while "Sun (NN)" only shares biLSTM hidden states. However, our "NN" performs within 0.6 point on relations. One possible reason might be that we do not use the features of output entity type for relation type classification.

2. After introducing graph convolutional networks, all three GCN-based models improve performances of entity and relation. Specifically, The "GCN (static)" has been slightly improved on relations. The "GCN (dynamic + soft)" achieves 0.7 percent improvement on relations and has the same entity performance. The "GCN (dynamic + hard)" improves the entity performance (0.4 percent) [6] and achieves large improvement (1.9 percent) in relation performance. It is competitive with state-of-the-art model (Sun et al., 2018). These observations show that the proposed joint model is effective for the joint type inference on entities and relations, and also show the rationality of the proposed dynamic graph, as expected.

3. The performances of the entity span and the binary relation are close to all proposed models. One possible reason is that there are more coarse-gained task. Effective features can be easily extracted for all models. It is worth noting that the performance in binary relation is not very good. Our dynamic graph relies on binary relation detection task. How to improve the performance of binary relation is still a hard question. We leave it as future work.

Thirdly, we present the influences of the number of GCN layers (Table 2). We take the "GCN (dynamic + hard)" as a example. In general, the performances on four tasks are insensitive to the number of GCN layers [7]. In particular, the performances on entity span, entity and relation fluctuate at 1.0 points, and the binary relation fluctuate at 1.4 points. Interestingly, we find the one layer GCN achieves best relation performance though the performances of other three tasks are not best. One possible reason is that the all models are closely related to each other. However, how they

[6] In fact, the entity performance on the ACE05 test data is hard to improve from past works (Miwa and Bansal, 2016; Zhang et al., 2017; Sun et al., 2018). So it is a non-negligible improvement over existing state-of-the-art systems.

[7] We focus on the performance of the end-to-end relation extraction, so we select models by the relation extraction results. It is also possible to consider both the performances of the entity model and the relation model. We leave the study of advanced model selection algorithms for future work.

Model	Entity			Relation			Entity Span			Binary Relation		
	P	R	F	P	R	F	P	R	F	P	R	F
Sun (NN) (2018)	84.0	**82.9**	83.4	59.5	**56.3**	57.8	-	-	-	-	-	-
NN	85.7	82.1	83.9	65.6	50.7	57.2	**91.2**	89.6	90.4	-	-	-
GCN (static)	85.0	82.6	83.8	66.6	51.3	57.8	90.8	**90.2**	**90.5**	-	-	-
GCN (dynamic + soft)	85.3	82.3	83.8	67.3	51.6	58.5	90.8	**90.2**	**90.5**	77.3	**56.4**	65.2
GCN (dynamic + hard)	**86.1**	82.4	**84.2**	**68.1**	52.3	**59.1**	**91.2**	89.5	90.4	**78.2**	56.3	**65.4**

Table 3: Results on the ACE05 dataset in different settings.

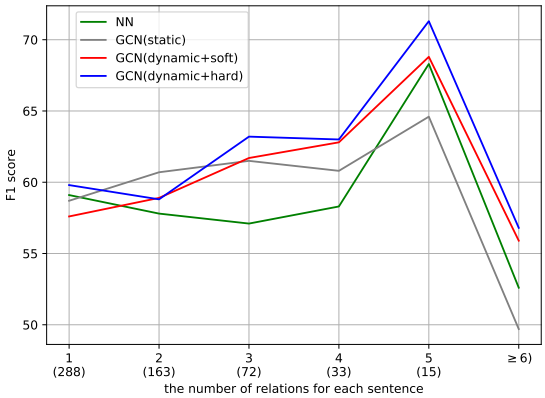

Figure 5: F1 scores with respect to the number of relations for each sentence. The numbers in parentheses are counts of sentences in the ACE05 test set.

Model	Relation		
M&B (2016)	**70.1**	61.2	65.3
C&M (2018)	69.7	59.5	64.2
NN	68.5	62.8	65.5
GCN (static)	69.1	63.8	66.4
GCN (dynamic + soft)	68.7	63.4	65.9
GCN (dynamic + hard)	68.7	**65.4**	**67.0**

Table 4: Results on the ACE05 dataset with golden entity.

affect each other in this joint settings is still an open question.

Forthly, we examine the relation performance with respect to different the number of relations for each sentence (Figure 5). In general, our GCN-based models almost outperform "NN" when the number of relations is larger than 2. It proves that the proposed GCN-based models are more suitable for handle multiple relations in a sentence. We think our method will perform better on the complex multiple relations dataset which is very common in reality.

Finally, We compare the "NN" model with the "GCN" model on some concrete examples, as shown in Table 5. For S1, the "NN" wrongly identifies the relation GEN-AFF between "[legislature]$^{\text{ORG}}$" and "[north korea]$^{\text{GPE}}$" even though the relation ORG-AFF between "[legislature]$^{\text{ORG}}$" and "[chairman]$^{\text{PER}}$" is detected. For S2, the "NN" does not detect PART-WHOLE relation while the "GCN" correctly find it. These two observations show that our "GCN" is good at dealing with the situation when the multiple relations share common entities, as expected. For S3, our "GCN" identifies a PHYS relation between "[units]$^{\text{PER}}$" and "[captial]$^{\text{GPE}}$", while the "NN" does not find this relation even the entities are correct. However, both models do not identify the relation ART between "[units]$^{\text{PER}}$" and "[weapons]$^{\text{WEA}}$". We think advanced improvement methods which use more powerful graph neural network might be helpful in this situation.

4.2 Golden Entity Results on ACE05

In order to compare with relation classification methods, we evaluate our models with golden entities on ACE05 corpus in Table 4. We use the same data split to compare with their model (Miwa and Bansal, 2016; Christopoulou et al., 2018). We do not tune hyperparameters extensively. For example, we use the same setting in both end-to-end and golden entity rather than tune parameters on each of them. The baseline systems are (Miwa and Bansal, 2016) and (Christopoulou et al., 2018).

In general, our "NN" is competitive, comparing to the dependency tree-based state-of-the-art model (Miwa and Bansal, 2016). It shows that our CNN-based neural networks are able to extract more powerful features to help relation extraction task. After adding GCN, our GCN-based models achieve the better performance. This indicates that the proposed models can achieve large improvement without any external syntactic tools [8].

[8]For simplicity, we do not extract golden entity type features explicitly in our model. And we believe there will be further improvements when these features are used.

S1	the [british]$^{\text{GPE:}\heartsuit\clubsuit\spadesuit}_{\text{GEN-AFF-2:}\heartsuit\clubsuit\spadesuit}$ [arm]$^{\text{ORG:}\heartsuit\clubsuit\spadesuit}_{\text{PART-WHOLE-1:}\heartsuit\spadesuit\mid\text{GEN-AFF-1:}\heartsuit\clubsuit\spadesuit}$ of french distributors [pathe]$^{\text{ORG:}\heartsuit\clubsuit\spadesuit}_{\text{PART-WHOLE-2:}\heartsuit\spadesuit}$ to show four releases .
S2	... [chairman]$^{\text{PER:}\heartsuit\clubsuit\spadesuit}_{\text{ORG-AFF-1:}\heartsuit\clubsuit\spadesuit}$ of [north korea]$^{\text{GPE:}\heartsuit\clubsuit\spadesuit}_{\text{PART-WHOLE-2:}\heartsuit\spadesuit\mid\text{GEN-AFF-2:}\clubsuit}$'s [legislature]$^{\text{ORG:}\heartsuit\clubsuit\spadesuit}_{\text{PART-WHOLE-1:}\heartsuit\spadesuit\mid\text{ORG-AFF-2:}\heartsuit\clubsuit\spadesuit\mid\text{GEN-AFF-1:}\clubsuit}$, the supreme people 's assembly .
S3	a red line may have been drawn around the [capital]$^{\text{GPE:}\heartsuit\clubsuit\spadesuit}_{\text{PHYS-2:}\heartsuit\spadesuit}$ with [republican gurad]$^{\text{ORG:}\heartsuit\clubsuit\spadesuit}_{\text{ORG-AFF-2:}\heartsuit\clubsuit\spadesuit}$ [units]$^{\text{PER:}\heartsuit\clubsuit\spadesuit}_{\text{PHYS-1:}\heartsuit\spadesuit\mid\text{ORG-AFF-1:}\heartsuit\clubsuit\spadesuit\mid\text{ART-1:}\heartsuit}$ ordered to use chemical [weapons]$^{\text{WEA:}\heartsuit\clubsuit\spadesuit}_{\text{ART-2:}\heartsuit}$ once u.s. and allied troops cross it .

Table 5: Examples from the ACE05 dataset with label annotations from "NN" model and "GCN" model for comparison. The ♡ is the gold standard, and the ♣, ♠ are the output of the "NN" ,"GCN" model respectively.

5 Related Work

There have been extensive studies for entity relation extraction task. Early work employs a pipeline of methods that extracts entities first, and then determines their relations (Zelenko et al., 2003; Miwa et al., 2009; Chan and Roth, 2011; Lin et al., 2016). As pipeline approaches suffer from error propagation, researchers have proposed methods for joint entity relation extraction.

Parameter sharing is a basic strategy for joint extraction. For example, Miwa and Bansal (2016) propose a neural method comprised of a sentence-level RNN for extracting entities, and a dependency tree-based RNN to predict relations. Their relation model takes hidden states of the entity model as features (i.e., the shared parameters). Similarly, Katiyar and Cardie (2017) use a simplified relation model based on the entity RNN using the attention mechanism. These joint methods do joint learning through sharing parameters and they have no explicit interaction in type inference.

To further explore interactions between the entity decoder and the relation decoder, many of them focus on some joint decoding algorithms. ILP-based joint decoder (Yang and Cardie, 2013), CRF-based joint decoder (Katiyar and Cardie, 2016), joint sequence labelling tag set (Zheng et al., 2017), beam search (Li and Ji, 2014), global normalization (Zhang et al., 2017), and transition system (Wang et al., 2018) are investigated. Different from models there, we propose a novel and concise joint model to handle joint type inference based on graph convolutional networks, which can capture information between multiple entity types and relation types explicitly[9].

Recently, researches of graph neural networks (GNNs) have been receiving more and more attention because of the great expressive power of graphs (Cai et al., 2018; Battaglia et al., 2018; Zhou et al., 2018). Graph Convolutional Network (GCN) is one of the typical variants of GNN (Bruna et al., 2013; Defferrard et al., 2016; Kipf and Welling, 2017). It has been successfully applied to many NLP tasks such as text classification (Yao et al., 2018), semantic role labeling (Marcheggiani and Titov, 2017), relation extraction (Zhang et al., 2018) machine translation (Bastings et al., 2017) and knowledge base completion (Shang et al., 2018). We note that most previous applications of GCN focus on a single job, while the joint entity relation extraction consists of multiple sub-tasks. Investigating GCN in joint learning scenarios is the main topic of this work. A closely related work is (Christopoulou et al., 2018), which focuses on relation extraction with golden entities. Our work can be viewed as an end-to-end extension of their work.

6 Conclusion

We propose a novel and concise joint model based on GCN to perform joint type inference for entity relation extraction task. Compared with existing joint methods, it provides a new way to capture the interactions on multiple entity types and relation types explicitly in a sentence. Experiments on ACE05 dataset show the effectiveness of the proposed method.

[9] In addition, transfer learning(Sun and Wu, 2019), multi-task learning (Sanh et al., 2018) for this task were also studied. In order to make a fair comparison, we do not include these models in experiments.

Acknowledgement

The authors wish to thank the reviewers for their helpful comments and suggestions. This research is (partially) supported by STCSM (18ZR1411500), NSFC(61673179) and the Foundation of State Key Laboratory of Cognitive Intelligence, iFLYTEK(COGOS-20190003). The corresponding authors are Yuanbin Wu and Shiliang Sun.

References

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 551–560, Portland, Oregon, USA. Association for Computational Linguistics.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2018. A walk-based model on entity graphs for relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 81–88. Association for Computational Linguistics.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 919–929, Berlin, Germany. Association for Computational Linguistics.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 917–928, Vancouver, Canada. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

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

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for 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*, pages 1506–1515.

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 (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 121–130, Singapore. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2018. A hierarchical multi-task approach for learning embeddings from semantic tasks. *arXiv preprint arXiv:1811.06031*.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2018. End-to-end structure-aware convolutional networks for knowledge base completion. *arXiv preprint arXiv:1811.04441*.

Changzhi Sun and Yuanbin Wu. 2019. Distantly supervised entity relation extraction with adapted manual annotations. In *Thirty-Third AAAI Conference on Artificial Intelligence*.

Changzhi Sun, Yuanbin Wu, Man Lan, Shiliang Sun, Wenting Wang, Kuang-Chih Lee, and Kewen Wu. 2018. Extracting entities and relations with joint minimum risk training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2265. Association for Computational Linguistics.

Shaolei Wang, Yue Zhang, Wanxiang Che, and Ting Liu. 2018. Joint extraction of entities and relations based on a novel graph scheme. In *IJCAI*, pages 4461–4467.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1640–1649.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. Graph convolutional networks for text classification. *arXiv preprint arXiv:1809.05679*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

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

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1731–1741, Copenhagen, Denmark. 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*, pages 2205–2215.

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 the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, Vancouver, Canada. Association for Computational Linguistics.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.