

# Extracting Relational Facts by an End-to-End Neural Model with Copy Mechanism

Xiangrong Zeng<sup>1,2</sup>, Daojian Zeng<sup>3</sup>, Shizhu He<sup>1</sup>, Kang Liu<sup>1,2</sup>, Jun Zhao<sup>1,2</sup>

<sup>1</sup>National Laboratory of Pattern Recognition (NLPR), Institute of Automation  
Chinese Academy of Sciences, Beijing, 100190, China

<sup>2</sup>University of Chinese Academy of Sciences, Beijing, 100049, China

<sup>3</sup>Changsha University of Science & Technology, Changsha, 410114, China  
{xiangrong.zeng, shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn  
zengdj@csust.edu.cn

## Abstract

The relational facts in sentences are often complicated. Different relational triplets may have overlaps in a sentence. We divided the sentences into three types according to triplet overlap degree, including *Normal*, *EntityPairOverlap* and *SingleEntityOverlap*. Existing methods mainly focus on *Normal* class and fail to extract relational triplets precisely. In this paper, we propose an end-to-end model based on sequence-to-sequence learning with copy mechanism, which can jointly extract relational facts from sentences of any of these classes. We adopt two different strategies in decoding process: employing only one united decoder or applying multiple separated decoders. We test our models in two public datasets and our model outperform the baseline method significantly.

## 1 Introduction

Recently, to build large structural knowledge bases (KB), great efforts have been made on extracting relational facts from natural language texts. A relational fact is often represented as a triplet which consists of two entities (an entity pair) and a semantic relation between them, such as  $\langle \textit{Chicago}, \textit{country}, \textit{UnitedStates} \rangle$ .

So far, most previous methods mainly focused on the task of relation extraction or classification which identifies the semantic relations between two pre-assigned entities. Although great progresses have been made (Hendrickx et al., 2010; Zeng et al., 2014; Xu et al., 2015a,b), they all assume that the entities are identified beforehand and neglect the extraction of entities. To extract both of entities and relations, early works (Zelenko et al., 2003; Chan and Roth, 2011) adopted a pipeline


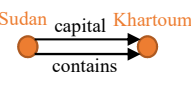
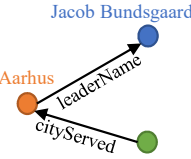
Normal	S1: <i>Chicago</i> is located in the <i>United States</i> .	
	{< <i>Chicago</i> , <i>country</i> , <i>United States</i> >}	
EPO	S2: News of the list's existence unnerved officials in <i>Khartoum</i> , <i>Sudan</i> 's capital.	
	{< <i>Sudan</i> , <i>capital</i> , <i>Khartoum</i> >, < <i>Sudan</i> , <i>contains</i> , <i>Khartoum</i> >}	
SEO	S3: <i>Aarhus airport</i> serves the city of <i>Aarhus</i> who's leader is <i>Jacob Bundsgaard</i> .	
	{< <i>Aarhus</i> , <i>leaderName</i> , <i>Jacob Bundsgaard</i> >, < <i>Aarhus Airport</i> , <i>cityServed</i> , <i>Aarhus</i> >}	

Figure 1: Examples of *Normal*, *EntityPairOverlap* (EPO) and *SingleEntityOverlap* (SEO) classes. The overlapped entities are marked in yellow. S1 belongs to *Normal* class because none of its triplets have overlapped entities; S2 belongs to *EntityPairOverlap* class since the entity pair  $\langle \textit{Sudan}, \textit{Khartoum} \rangle$  of its two triplets are overlapped; And S3 belongs to *SingleEntityOverlap* class because the entity *Aarhus* of its two triplets are overlapped and these two triplets have no overlapped entity pair.

manner, where they first conduct entity recognition and then predict relations between extracted entities. However, the pipeline framework ignores the relevance of entity identification and relation prediction (Li and Ji, 2014). Recent works attempted to extract entities and relations jointly. Yu and Lam (2010); Li and Ji (2014); Miwa and Sasaki (2014) designed several elaborate features to construct the bridge between these two subtasks. Similar to other natural language processing (NLP) tasks, they need complicated feature engineering and heavily rely on pre-existing NLP tools for feature extraction.

Recently, with the success of deep learning on many NLP tasks, it is also applied on relational facts extraction. Zeng et al. (2014); Xu et al. (2015a,b) employed CNN or RNN on relation classification. Miwa and Bansal (2016); Gupta et al. (2016); Zhang et al. (2017) treated relation extraction task as an end-to-end (end2end) table-filling problem. Zheng et al. (2017) proposed a novel tagging schema and employed a Recurrent Neural Networks (RNN) based sequence labeling model to jointly extract entities and relations.

Nevertheless, the relational facts in sentences are often complicated. Different relational triplets may have overlaps in a sentence. Such phenomenon makes aforementioned methods, whatever deep learning based models and traditional feature engineering based joint models, always fail to extract relational triplets precisely. Generally, according to our observation, we divide the sentences into three types according to triplet overlap degree, including *Normal*, *EntityPairOverlap* (EPO) and *SingleEntityOverlap* (SEO). As shown in Figure 1, a sentence belongs to *Normal* class if none of its triplets have overlapped entities. A sentence belongs to *EntityPairOverlap* class if some of its triplets have overlapped entity pair. And a sentence belongs to *SingleEntityOverlap* class if some of its triplets have an overlapped entity and these triplets don't have overlapped entity pair. In our knowledge, most previous methods focused on *Normal* type and seldom consider other types. Even the joint models based on neural network (Zheng et al., 2017), it only assigns a single tag to a word, which means one word can only participate in at most one triplet. As a result, the triplet overlap issue is not actually addressed.

To address the aforementioned challenge, we aim to design a model that could extract triplets, including entities and relations, from sentences of *Normal*, *EntityPairOverlap* and *SingleEntityOverlap* classes. To handle the problem of triplet overlap, one entity must be allowed to freely participate in multiple triplets. Different from previous neural methods, we propose an end2end model based on sequence-to-sequence (Seq2Seq) learning with copy mechanism, which can jointly extract relational facts from sentences of any of these classes. Specially, the main component of this model includes two parts: encoder and decoder. The encoder converts a natural language sentence (the source sentence) into a fixed length semantic

vector. Then, the decoder reads in this vector and generates triplets directly. To generate a triplet, firstly, the decoder generates the relation. Secondly, by adopting the copy mechanism, the decoder copies the first entity (head entity) from the source sentence. Lastly, the decoder copies the second entity (tail entity) from the source sentence. In this way, multiple triplets can be extracted (In detail, we adopt two different strategies in decoding process: employing only one unified decoder (OneDecoder) to generate all triplets or applying multiple separated decoders (MultiDecoder) and each of them generating one triplet). In our model, one entity is allowed to be copied several times when it needs to participate in different triplets. Therefore, our model could handle the triplet overlap issue and deal with both of *EntityPairOverlap* and *SingleEntityOverlap* sentence types. Moreover, since extracting entities and relations in a single end2end neural network, our model could extract entities and relations jointly.

The main contributions of our work are as follows:

- We propose an end2end neural model based on sequence-to-sequence learning with copy mechanism to extract relational facts from sentences, where the entities and relations could be jointly extracted.
- Our model could consider the relational triplet overlap problem through copy mechanism. In our knowledge, the relational triplet overlap problem has never been addressed before.
- We conduct experiments on two public datasets. Experimental results show that we outperforms the state-of-the-arts with 39.8% and 31.1% improvements respectively.

## 2 Related Work

By giving a sentence with annotated entities, Hendrickx et al. (2010); Zeng et al. (2014); Xu et al. (2015a,b) treat identifying relations in sentences as a multi-class classification problem. Zeng et al. (2014) among the first to introduce CNN into relation classification. Xu et al. (2015a) and Xu et al. (2015b) learned relation representations from shortest dependency paths through a CNN or RNN. Despite their success, these models ignore the extraction of the entities from sentences and could not truly extract relational facts.

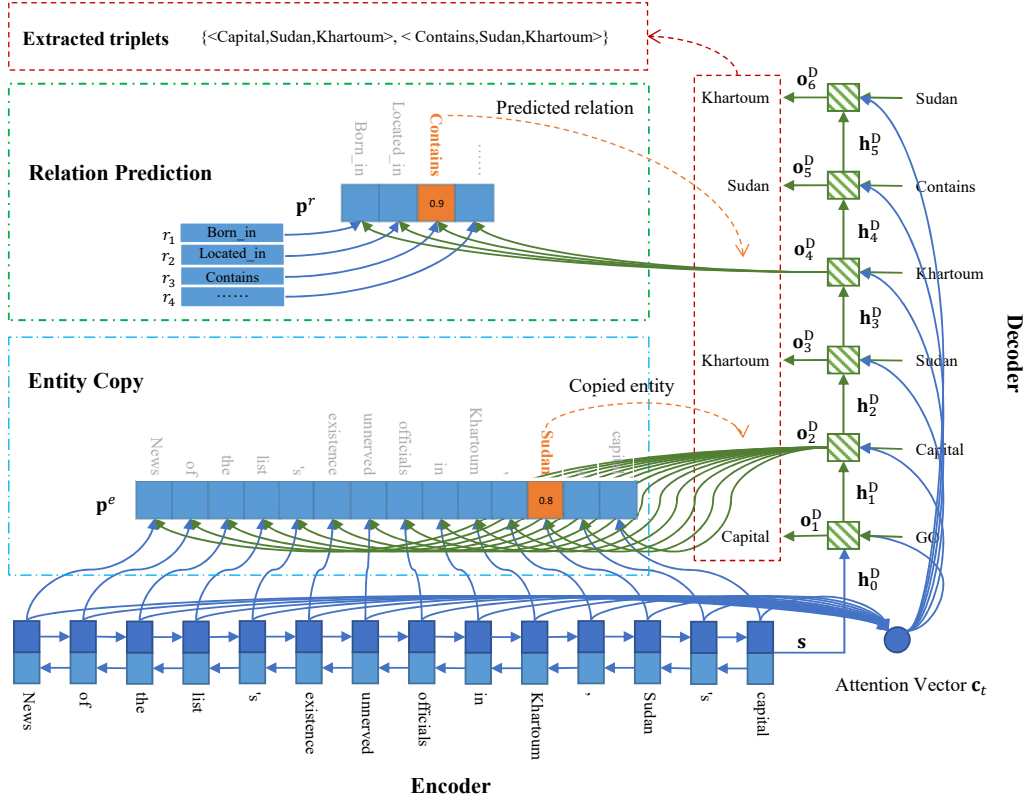


Figure 2: The overall structure of OneDecoder model. A bi-directional RNN is used to encode the source sentence and then a decoder is used to generate triples directly. The relation is predicted and the entity is copied from source sentence.

By giving a sentence without any annotated entities, researchers proposed several methods to extract both entities and relations. Pipeline based methods, like Zelenko et al. (2003) and Chan and Roth (2011), neglected the relevance of entity extraction and relation prediction. To resolve this problem, several joint models have been proposed. Early works (Yu and Lam, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014) need complicated process of feature engineering and heavily depends on NLP tools for feature extraction. Recent models, like Miwa and Bansal (2016); Gupta et al. (2016); Zhang et al. (2017); Zheng et al. (2017), jointly extract the entities and relations based on neural networks. These models are based on tagging framework, which assigns a relational tag to a word or a word pair. Despite their success, none of these models can fully handle the triplet overlap problem mentioned in the first section. The reason is in their hypothesis, that is, a word (or a word pair) can only be assigned with just one relational tag.

This work is based on sequence-to-sequence learning with copy mechanism, which have been

adopted for some NLP tasks. Dong and Lapata (2016) presented a method based on an attention-enhanced and encoder-decoder model, which encodes input utterances and generates their logical forms. Gu et al. (2016); He et al. (2017) applied copy mechanism to sentence generation. They copy a segment from the source sequence to the target sequence.

### 3 Our Model

In this section, we introduce a differentiable neural model based on Seq2Seq learning with copy mechanism, which is able to extract multiple relational facts in an end2end fashion.

Our neural model encodes a variable-length sentence into a fixed-length vector representation first and then decodes this vector into the corresponding relational facts (triplets). When decoding, we can either decode all triplets with one unified decoder or decode every triplet with a separated decoder. We denote them as **OneDecoder** model and **MultiDecoder** model separately.

### 3.1 OneDecoder Model

The overall structure of OneDecoder model is shown in Figure 2.

#### 3.1.1 Encoder

To encode a sentence  $s = [w_1, \dots, w_n]$ , where  $w_t$  represent the  $t$ -th word and  $n$  is the source sentence length, we first turn it into a matrix  $X = [x_1, \dots, x_n]$ , where  $x_t$  is the embedding of  $t$ -th word.

The canonical RNN encoder reads this matrix  $X$  sequentially and generates output  $\mathbf{o}_t^E$  and hidden state  $\mathbf{h}_t^E$  in time step  $t$  ( $1 \leq t \leq n$ ) by

$$\mathbf{o}_t^E, \mathbf{h}_t^E = f(x_t, \mathbf{h}_{t-1}^E) \quad (1)$$

where  $f(\cdot)$  represents the encoder function.

Following (Gu et al., 2016), our encoder uses a bi-directional RNN (Chung et al., 2014) to encode the input sentence. The forward and backward RNN obtain output sequence  $\{\overrightarrow{\mathbf{o}}_1^E, \dots, \overrightarrow{\mathbf{o}}_n^E\}$  and  $\{\overleftarrow{\mathbf{o}}_n^E, \dots, \overleftarrow{\mathbf{o}}_1^E\}$ , respectively. We then concatenate  $\overrightarrow{\mathbf{o}}_t^E$  and  $\overleftarrow{\mathbf{o}}_{n-t+1}^E$  to represent the  $t$ -th word. We use  $\mathbf{O}^E = [\mathbf{o}_1^E, \dots, \mathbf{o}_n^E]$ , where  $\mathbf{o}_t^E = [\overrightarrow{\mathbf{o}}_t^E; \overleftarrow{\mathbf{o}}_{n-t+1}^E]$ , to represent the concatenate result. Similarly, the concatenation of forward and backward RNN hidden states are used as the representation of sentence, that is  $\mathbf{s} = [\overrightarrow{\mathbf{h}}_n^E; \overleftarrow{\mathbf{h}}_1^E]$

#### 3.1.2 Decoder

The decoder is used to generate triplets directly. Firstly, the decoder generates a relation for the triplet. Secondly, the decoder copies an entity from the source sentence as the first entity of the triplet. Lastly, the decoder copies the second entity from the source sentence. Repeat this process, the decoder could generate multiple triplets. Once all valid triplets are generated, the decoder will generate NA triplets, which means ‘‘stopping’’ and is similar to the ‘‘eos’’ symbol in neural sentence generation. Note that, a NA triplet is composed of an NA-relation and an NA-entity pair.

As shown in Figure 3 (a), in time step  $t$  ( $1 \leq t$ ), we calculate the decoder output  $\mathbf{o}_t^D$  and hidden state  $\mathbf{h}_t^D$  as follows:

$$\mathbf{o}_t^D, \mathbf{h}_t^D = g(\mathbf{u}_t, \mathbf{h}_{t-1}^D) \quad (2)$$

where  $g(\cdot)$  is the decoder function and  $\mathbf{h}_{t-1}^D$  is the hidden state of time step  $t - 1$ . We initialize  $\mathbf{h}_0^D$  with the representation of source sentence  $\mathbf{s}$ .  $\mathbf{u}_t$  is

the decoder input in time step  $t$  and we calculate it as:

$$\mathbf{u}_t = [\mathbf{v}_t; \mathbf{c}_t] \cdot \mathbf{W}^u \quad (3)$$

where  $\mathbf{c}_t$  is the attention vector and  $\mathbf{v}_t$  is the embedding of copied entity or predicted relation in time step  $t - 1$ .  $\mathbf{W}^u$  is a weight matrix.

**Attention Vector.** The attention vector  $\mathbf{c}_t$  is calculated as follows:

$$\mathbf{c}_t = \sum_{i=1}^n \alpha_i \times \mathbf{o}_i^E \quad (4)$$

$$\alpha = \text{softmax}(\beta) \quad (5)$$

$$\beta_i = \text{selu}([\mathbf{h}_{t-1}^D; \mathbf{o}_i^E] \cdot \mathbf{w}^c) \quad (6)$$

where  $\mathbf{o}_i^E$  is the output of encoder in time step  $i$ ,  $\alpha = [\alpha_1, \dots, \alpha_n]$  and  $\beta = [\beta_1, \dots, \beta_n]$  are vectors,  $\mathbf{w}^c$  is a weight vector.  $\text{selu}(\cdot)$  is activation function (Klambauer et al., 2017).

After we get decoder output  $\mathbf{o}_t^D$  in time step  $t$  ( $1 \leq t$ ), if  $t\%3 = 1$  (that is  $t = 1, 4, 7, \dots$ ), we use  $\mathbf{o}_t^D$  to predict a relation, which means we are decoding a new triplet. Otherwise, if  $t\%3 = 2$  (that is  $t = 2, 5, 8, \dots$ ), we use  $\mathbf{o}_t^D$  to copy the first entity from the source sentence, and if  $t\%3 = 0$  (that is  $t = 3, 6, 9, \dots$ ), we copy the second entity.

**Predict Relation.** Suppose there are  $m$  valid relations in total. We use a fully connected layer to calculate the confidence vector  $\mathbf{q}^r = [q_1^r, \dots, q_m^r]$  of all valid relations:

$$\mathbf{q}^r = \text{selu}(\mathbf{o}_t^D \cdot \mathbf{W}^r + \mathbf{b}^r) \quad (7)$$

where  $\mathbf{W}^r$  is the weight matrix and  $\mathbf{b}^r$  is the bias. When predict the relation, it is possible to predict the NA-relation when the model try to generate NA-triplet. To take this into consideration, we calculate the confidence value of NA-relation as:

$$\mathbf{q}^{NA} = \text{selu}(\mathbf{o}_t^D \cdot \mathbf{W}^{NA} + \mathbf{b}^{NA}) \quad (8)$$

where  $\mathbf{W}^{NA}$  is the weight matrix and  $\mathbf{b}^{NA}$  is the bias. We then concatenate  $\mathbf{q}^r$  and  $\mathbf{q}^{NA}$  to form the confidence vector of all relations (including the NA-relation) and apply softmax to obtain the probability distribution  $\mathbf{p}^r = [p_1^r, \dots, p_{m+1}^r]$  as:

$$\mathbf{p}^r = \text{softmax}([\mathbf{q}^r; \mathbf{q}^{NA}]) \quad (9)$$

We select the relation with the highest probability as the predict relation and use it’s embedding as the next time step input  $\mathbf{v}_{t+1}$ .

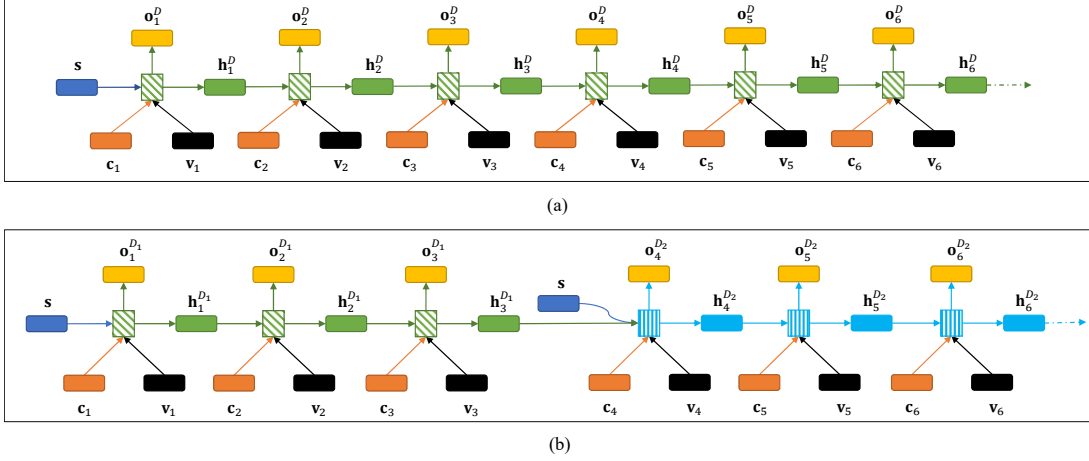


Figure 3: The inputs and outputs of the decoder(s) of OneDecoder model and MultiDecoder model. (a) is the decoder of OneDecoder model. As we can see, only one decoder (the green rectangle with shadows) is used and this encoder is initialized with the sentence representation  $\mathbf{s}$ . (b) is the decoders of MultiDecoder model. There are two decoders (the green rectangle and blue rectangle with shadows). The first decoder is initialized with  $\mathbf{s}$ ; Other decoder(s) are initialized with  $\mathbf{s}$  and previous decoder’s state.

**Copy the First Entity.** To copy the first entity, we calculate the confidence vector  $\mathbf{q}^e = [q_1^e, \dots, q_n^e]$  of all words in source sentence as:

$$q_i^e = \text{selu}([\mathbf{o}_t^D; \mathbf{o}_i^E] \cdot \mathbf{w}^e) \quad (10)$$

where  $\mathbf{w}^e$  is the weight vector. Similar with the relation prediction, we concatenate  $\mathbf{q}^e$  and  $\mathbf{q}^{NA}$  to form the confidence vector and apply softmax to obtain the probability distribution  $\mathbf{p}^e = [p_1^e, \dots, p_{n+1}^e]$ :

$$\mathbf{p}^e = \text{softmax}([\mathbf{q}^e; \mathbf{q}^{NA}]) \quad (11)$$

Similarly, We select the word with the highest probability as the predict the word and use it’s embedding as the next time step input  $\mathbf{v}_{t+1}$ .

**Copy the Second Entity.** Copy the second entity is almost the same as copy the first entity. The only difference is when copying the second entity, we cannot copy the first entity again. This is because in a valid triplet, two entities must be different. Suppose the first copied entity is the  $k$ -th word in the source sentence, we introduce a mask vector  $M$  with  $n$  ( $n$  is the length of source sentence) elements, where:

$$M_i = \begin{cases} 1, & i \neq k \\ 0, & i = k \end{cases} \quad (12)$$

then we calculate the probability distribution  $\mathbf{p}^e$  as:

$$\mathbf{p}^e = \text{softmax}([M \otimes \mathbf{q}^e; \mathbf{q}^{NA}]) \quad (13)$$

where  $\otimes$  is element-wise multiplication. Just like copy the first entity, We select the word with the highest probability as the predict word and use it’s embedding as the next time step input  $\mathbf{v}_{t+1}$ .

### 3.2 MultiDecoder Model

MultiDecoder model is an extension of the proposed OneDecoder model. The main difference is when decoding triplets, MultiDecoder model decode triplets with several separated decoders. Figure 3 (b) shows the inputs and outputs of decoders of MultiDecoder model. There are two decoders (the green and blue rectangle with shadows). Decoders work in a sequential order: the first decoder generate the first triplet and then the second decoder generate the second triplet.

Similar with Eq 2, we calculate the hidden state  $\mathbf{h}_t^{D_i}$  and output  $\mathbf{o}_t^{D_i}$  of  $i$ -th ( $1 \leq i$ ) decoder in time step  $t$  as follows:

$$\mathbf{o}_t^{D_i}, \mathbf{h}_t^{D_i} = \begin{cases} g^{D_i}(\mathbf{u}_t, \mathbf{h}_{t-1}^{D_i}), & t \% 3 = 2, 0 \\ g^{D_i}(\mathbf{u}_t, \hat{\mathbf{h}}_{t-1}^{D_i}), & t \% 3 = 1 \end{cases} \quad (14)$$

$g^{D_i}(\cdot)$  is the decoder function of decoder  $i$ .  $\mathbf{u}_t$  is the decoder input in time step  $t$  and we calculated it as Eq 3.  $\mathbf{h}_{t-1}^{D_i}$  is the hidden state of  $i$ -th decoder in time step  $t - 1$ .  $\hat{\mathbf{h}}_{t-1}^{D_i}$  is the initial hidden state of  $i$ -th decoder, which is calculated as follows:

$$\hat{\mathbf{h}}_{t-1}^{D_i} = \begin{cases} \mathbf{s}, & i = 1 \\ \frac{1}{2}(\mathbf{s} + \mathbf{h}_{t-1}^{D_{i-1}}), & i > 1 \end{cases} \quad (15)$$



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

Table 1: The number of sentences of *Normal*, *EntityPairOverlap* (*EPO*) and *SingleEntityOverlap* (*SEO*) classes. It’s worthy noting that a sentence can belongs to both *EPO* class and *SEO* class.

### 3.3 Training

Both OneDecoder and MultiDecoder models are trained with the negative log-likelihood loss function. Given a batch of data with  $B$  sentences  $S = \{s_1, \dots, s_B\}$  with the target results  $Y = \{\mathbf{y}_1, \dots, \mathbf{y}_B\}$ , where  $\mathbf{y}_i = [y_i^1, \dots, y_i^T]$  is the target result of  $s_i$ , the loss function is defined as follows:

$$\mathbf{L} = \frac{1}{B \times T} \sum_{i=1}^B \sum_{t=1}^T -\log(p(y_i^t | y_i^{<t}, s_i, \theta)) \quad (16)$$

$T$  is the maximum time step of decoder.  $p(x|y)$  is the conditional probability of  $x$  given  $y$ .  $\theta$  denotes parameters of the entire model.

## 4 Experiments

### 4.1 Dataset

To evaluate the performance of our methods, we conduct experiments on two widely used datasets.

The first is New York Times (NYT) dataset, which is produced by distant supervision method (Riedel et al., 2010). This dataset consists of 1.18M sentences sampled from 294k 1987-2007 New York Times news articles. There are 24 valid relations in total. In this paper, we treat this dataset as supervised data as the same as Zheng et al. (2017). We filter the sentences with more than 100 words and the sentences containing no positive triplets, and 66195 sentences are left. We randomly select 5000 sentences from it as the test set, 5000 sentences as the validation set and the rest 56195 sentences are used as train set.

The second is WebNLG dataset (Gardent et al., 2017). It is originally created for Natural Language Generation (NLG) task. This dataset contains 246 valid relations. In this dataset, a instance including a group of triplets and several standard sentences (written by human). Every standard sentence contains all triplets of this instance. We on-

ly use the first standard sentence in our experiments and we filter out the instances if all entities of triplets are not found in this standard sentence. The origin WebNLG dataset contains train set and development set. In our experiments, we treat the origin development set as test set and randomly split the origin train set into validation set and train set. After filtering and splitting, the train set contains 5019 instances, the test set contains 703 instances and the validation set contains 500 instances.

The number of sentences of every class in NYT and WebNLG dataset are shown in Table 1. It’s worthy noting that a sentence can belongs to both *EntityPairOverlap* class and *SingleEntityOverlap* class.

### 4.2 Settings

In our experiments, for both dataset, we use LSTM (Hochreiter and Schmidhuber, 1997) as the model cell; The cell unit number is set to 1000; The embedding dimension is set to 100; The batch size is 100 and the learning rate is 0.001; The maximum time steps  $T$  is 15, which means we predict at most 5 triplets for each sentence (therefore, there are 5 decoders in MultiDecoder model). These hyperparameters are tuned on the validation set. We use Adam (Kingma and Ba, 2015) to optimize parameters and we stop the training when we find the best result in the validation set.

### 4.3 Baseline and Evaluation Metrics

We compare our models with NovelTagging model (Zheng et al., 2017), which conduct the best performance on relational facts extraction. We directly run the code released by Zheng et al. (2017) to acquire the results.

Following Zheng et al. (2017), we use the standard micro Precision, Recall and F1 score to evaluate the results. Triplets are regarded as correct when it’s relation and entities are both correct. When copying the entity, we only copy the last word of it. A triplet is regarded as NA-triplet when and only when it’s relation is NA-relation and it has an NA-entity pair. The predicted NA-triplets will be excluded.

### 4.4 Results

Table 2 shows the Precision, Recall and F1 value of NovelTagging model (Zheng et al., 2017) and our OneDecoder and MultiDecoder models.

Model	NYT			WebNLG		
	Precision	Recall	F1	Precision	Recall	F1
NovelTagging	<b>0.624</b>	0.317	0.420	<b>0.525</b>	0.193	0.283
OneDecoder	0.594	0.531	0.560	0.322	0.289	0.305
MultiDecoder	0.610	<b>0.566</b>	<b>0.587</b>	0.377	<b>0.364</b>	<b>0.371</b>

Table 2: Results of different models in NYT dataset and WebNLG dataset.

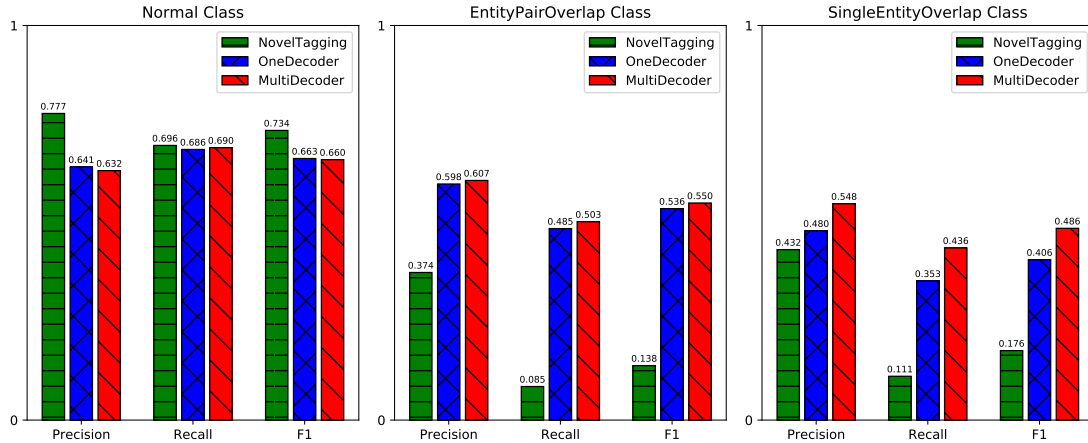


Figure 4: Results of NovelTagging, OneDecoder, and MultiDecoder model in *Normal*, *EntityPairOverlap* and *SingleEntityOverlap* classes in NYT dataset.

As we can see, in NYT dataset, our MultiDecoder model achieves the best F1 score, which is 0.587. There is 39.8% improvement compared with the NovelTagging model, which is 0.420. Besides, our OneDecoder model also outperforms the NovelTagging model. In the WebNLG dataset, MultiDecoder model achieves the highest F1 score (0.371). MultiDecoder and OneDecoder models outperform the NovelTagging model with 31.1% and 7.8% improvements, respectively. These observations verify the effectiveness of our models.

We can also observe that, in both NYT and WebNLG dataset, the NovelTagging model achieves the highest precision value and lowest recall value. By contrast, our models are much more balanced. We think that the reason is in the structure of the proposed models. The NovelTagging method finds triplets through tagging the words. However, they assume that only one tag could be assigned to just one word. As a result, one word can participate at most one triplet. Therefore, the NovelTagging model can only recall a small number of triplets, which harms the recall performance. Different from the NovelTagging model, our models apply copy mechanism to find entities for a triplet, and a word can be copied

many times when this word needs to participate in multiple different triplets. Not surprisingly, our models recall more triplets and achieve higher recall value. Further experiments verified this.

#### 4.5 Detailed Results on Different Sentence Types

To verify the ability of our models in handling the overlapping problem, we conduct further experiments on NYT dataset.

Figure 4 shows the results of NovelTagging, OneDecoder and MultiDecoder model in *Normal*, *EntityPairOverlap* and *SingleEntityOverlap* classes. As we can see, our proposed models perform much better than NovelTagging model in *EntityPairOverlap* class and *SingleEntityOverlap* classes. Specifically, our models achieve much higher performance on all metrics. Another observation is that NovelTagging model achieves the best performance in *Normal* class. This is because the NovelTagging model is designed more suitable for *Normal* class. However, our proposed models are more suitable for the triplet overlap issues. Furthermore, it is still difficult for our models to judge how many triplets are needed for the input sentence. As a result, there is a loss in our models for *Normal* class. Nevertheless, the overall perfor-

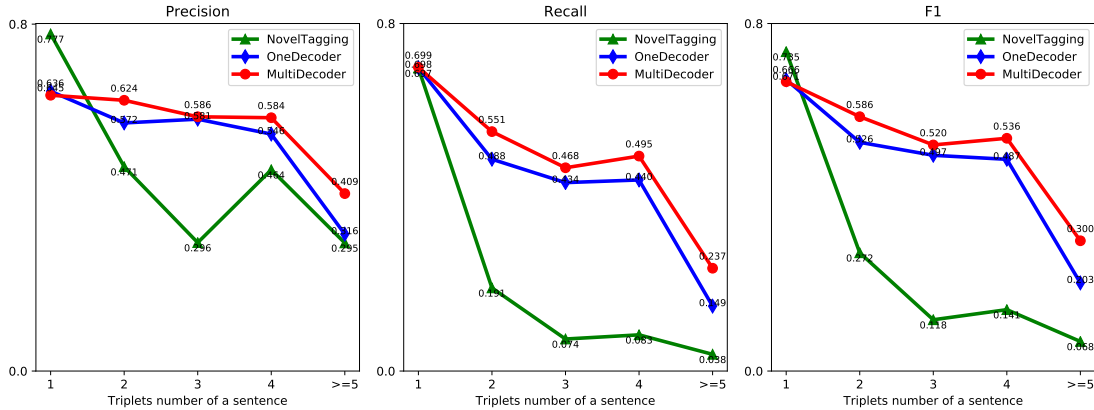


Figure 5: Relation Extraction from sentences that contains different number of triplets. We divide the sentences of NYT test set into 5 subclasses. Each class contains sentences that have 1,2,3,4 or  $\geq 5$  triplets.

Model	NYT	WebNLG
OneDecoder	0.858	0.745
MultiDecoder	0.862	0.821

Table 3: F1 values of entity generation.

Model	NYT	WebNLG
OneDecoder	0.874	0.759
MultiDecoder	0.870	0.751

Table 4: F1 values of relation generation.

mance of the proposed models still outperforms NoverTagging. Moreover, we notice that the whole extracted performance of *EntityPairOverlap* and *SingleEntityOverlap* class is lower than that in *Normal* class. It proves that extracting relational facts from *EntityPairOverlap* and *SingleEntityOverlap* classes are much more challenging than from *Normal* class.

We also compare the model’s ability of extracting relations from sentences that contains different number of triplets. We divide the sentences in NYT test set into 5 subclasses. Each class contains sentences that has 1,2,3,4 or  $\geq 5$  triplets. The results are shown in Figure 5. When extracting relation from sentences that contains 1 triplets, NovelTagging model achieve the best performance. However, when the number of triplets increases, the performance of NovelTagging model decreases significantly. We can also observe the huge decrease of recall value of NovelTagging model. These experimental results demonstrate the ability of our model in handling multiple relation extraction.

#### 4.6 OneDecoder vs. MultiDecoder

As shown in the previous experiments (Table 2, Figure 4 and Figure 5), our MultiDecoder model performs better then OneDecoder model and Nov-

elTagging model. To find out why MultiDecoder model performs better than OneDecoder model, we analyzed their ability of entity generation and relation generation. The experiment results are shown in Table 3 and Table 4. We can observe that on both NYT and WebNLG datasets, these two models have comparable abilities on relation generation. However, MultiDecoder performs better than OneDecoder model when generating entities. We think that it is because MultiDecoder model utilizes different decoder to generate different triplets so that the entity generation results could be more diverse.

### Conclusions and Future Work

In this paper, we proposed an end2end neural model based on Seq2Seq learning framework with copy mechanism for relational facts extraction. Our model can jointly extract relation and entity from sentences, especially when triplets in the sentences are overlapped. Moreover, we analyze the different overlap types and adopt two strategies for this issue, including one unified decoder and multiple separated decoders. We conduct experiments on two public datasets to evaluate the effectiveness of our models. The experiment results show that our models outperform the baseline method signif-



icantly and our models can extract relational facts from all three classes.

This challenging task is far from being solved. Our future work will concentrate on how to improve the performance further. Another future work is test our model in other NLP tasks like event extraction.

## Acknowledgments

The authors thank Yi Chang from Huawei Tech. Ltm for his helpful discussions. This work is supported by the Natural Science Foundation of China (No.61533018 and No.61702512). This work is also supported in part by Beijing Unisound Information Technology Co., Ltd, and Huawei Innovation Research Program of Huawei Tech. Ltm.

## References

- Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of ACL*, pages 551–560.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*, pages 33–43.
- Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planners. In *Proceedings of ACL*, pages 179–188.
- Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*, pages 1631–1640.
- Pankaj Gupta, Hinrich Schtze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING*, pages 2537–2547.
- Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of ACL*, pages 199–208.
- Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a Method for Stochastic Optimization. In *Proceedings of ICLR*, pages 1–15.
- Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in NIPS*, pages 971–980.
- Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of ACL*, pages 402–412.
- Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of ACL*, pages 1105–1116.
- Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of EMNLP*, pages 1858–1869.
- Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML PKDD*, pages 148–163.
- Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of EMNLP*, pages 536–540.
- Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of EMNLP*, pages 1785–1794.
- Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of COLING*, pages 1399–1407.
- Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106.
- Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.
- Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of EMNLP*, pages 1730–1740.
- 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*, pages 1227–1236.