Learning the Extraction Order of Multiple Relational Facts in a Sentence with Reinforcement Learning

Xiangrong Zeng[1,2,3], **Shizhu He**[1,2], **Daojian Zeng**[4], **Kang Liu**[1,2], **Jun Zhao**[1,2]
[1]NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
[3]Unisound AI Technology Co.,Ltd, Beijing, 100000, China
[4]Changsha University of Science & Technology, Changsha, 410114, China
{xiangrong.zeng, shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn
zengdj@csust.edu.cn

Abstract

The multiple relation extraction task tries to extract all relational facts from a sentence. Existing works didn't consider the extraction order of relational facts in a sentence. In this paper we argue that the extraction order is important in this task. To take the extraction order into consideration, we apply the reinforcement learning into a sequence-to-sequence model. The proposed model could generate relational facts freely. Widely conducted experiments on two public datasets demonstrate the efficacy of the proposed method.

1 Introduction

Relation extraction (RE) is a core task in natural language processing (NLP). RE can be used in information extraction (Wu and Weld, 2010), question answering (Yih et al., 2015; Dai et al., 2016) and other NLP tasks. Most existing works assumed that a sentence only contains one relational facts (a relational fact, or a triplet, contains a relation and two entities). But in fact, a sentence often contains multiple relational facts (Zeng et al., 2018b). The multiple relation extraction task tries to extract all relational facts from a sentence.

Existing works on multiple relation extraction task can be divided into five genres. 1) PseudoPipeline genre, including Miwa and Bansal (2016); Sun et al. (2018). They first recognized all the entities of the sentence, then extracted features for each entity pair and predicted their relation. They trained the entity recognition model and relation prediction model together instead of separately. Therefore, we call them PseudoPipeline methods. 2) TableFilling genre, including Miwa and Sasaki (2014); Gupta et al. (2016) and Zhang et al. (2017). They maintained a entity-relation table and predicted a semantic tag (either entity tags or relation tags) for each cell in the table. According to the predicted tags, they can recognize the entities and the relation between each entity pair. 3) NovelTagging genre, including Zheng et al. (2017). This method can be seen as a development of TableFilling method. They assigned a pre-defined semantic tag to each word of the sentence and collected triplets based on the tags. Their tags include both entity and relation information. Therefore, they don't need to maintain a entity-relation table. 4) MultiHeadSelection genre, including Bekoulis et al. (2018a) and Bekoulis et al. (2018b). They first recognized the entities, then they formulated the relation extraction task as a multi-head selection problem. For each entity, they calculated the score between it and every other entities for a given relation. The combination of the entity pair and relation with the score exceeding a threshold will be kept as a triplet. 5) Generative genre, including Zeng et al. (2018b). They directly generate triplets one by one by a sequence-to-sequence model with copy mechanism (Gu et al., 2016; Vinyals et al., 2015). To generate a triplet, they first generated the relation, then they copy the first entity and the second entity from the source sentence.

However, none of them have considered the extraction order of multiple triplets in a sentence. Given a sentence, the PseudoPipeline methods extract relations of different entity pairs separately. Although they jointly training the entity model and relation model, they ignore the influence between triplets actually. The TableFilling, NovelTagging and MultiHeadSelection methods extract the triplets in the word order of this sentence. They firstly deal with the first word, then the second one and so on. The generative method could generate triplets in any order actually. However, Zeng et al. (2018b) randomly choose the extraction order of the triplets in each sentence. Sorting the triplets in the sentences of training data beforehand with

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 367–377, Hong Kong, China, November 3–7, 2019.

Sentence	Cubanelle is in Arros negre, a dish from the Catalonia region.	
Relational Facts	#1	F_2: <Arros negre, food_region, Catalonia>
	#2	F_1: <Arros negre, ingredient, Cubanelle>

Figure 1: Example of multiple relation extraction. In this example, it's easier to extract F_2 first. The extraction of F_1 can be benefit from F_2.

global rules (e.g., alphabetical order) is straightforward. But one global sorting rule may not fit every sentences.

In this paper, we argue that the extraction order of triplets in a sentence is important. Take Figure 1 as example. It's difficult to extract F_1 first because we don't know what "Arros negre" is in the first place. Extracting F_2 is more straightforward as the key words "dish", "region" in the sentence is helpful. F_2 can help us to extract F_1 because now we are confident that "Arros negre" is some kind of food, so that "ingredient" is a suitable relation between "Arros negre" and "Cubanelle". From this intuitive example, we can see that the extracted triplets could influence the extraction of the remaining triplets.

To automatically learning the extraction order of multiple relational facts in a sentence, we propose a sequence-to-sequence model and apply reinforcement learning (RL) on it. we follow the Generative genre because such a model could extract triplets in various order, which is convenient for us to explore the influence of triplets extraction order. Our model reads in a raw sentence and generates triplets one by one. Thus, all triplets in a sentence could be extracted. To take the triplets extraction order into consideration, we convert the triplets generation process as a RL process. The sequence-to-sequence model is regarded as the RL policy. The action is what we generate in each time step. We assume that a better generation order could lead to more valid generated triplets. The RL reward is related to the generated triplets. In general, the more triplets are correctly generated, the higher the reward. Unlike supervised learning with negative log likelihood (NLL) loss, which forces the model to generate triplets in the order of the ground truth, reinforcement learning allows the model generate triplets freely to achieve higher reward.

The main contributions of this work are:

- We discuss the triplets extraction order problem in the multiple relation extraction task. In our knowledge, this problem has never been addressed before.
- We apply reinforcement learning method on a sequence-to-sequence model to handle this problem.
- We conduct widely experiments on two public datasets. Experimental results show that the proposed method outperform the strong baselines with 3.4% and 5.5% improvements respectively.

2 Related Work

Given a sentence with two annotated entities (an entity pair), the relation classification task aims to identify the predefined relation between these two entities. Zeng et al. (2014) was among the first to apply neural networks in relation classification task. They adopted the Convolutional Neural Network (CNN) to learn the sentence representation automatically. In the following, dos Santos et al. (2015); Xu et al. (2015a) also applied CNN to extract relation. Xu et al. (2015b) utilized shortest dependency path between two entities with a LSTM (Hochreiter and Schmidhuber, 1997) based recurrent neural network. Zhou et al. (2016) applied attention mechanism to learn different weights for each word and used LSTM to represent sentence. These methods all assumed that the entity pair is given beforehand and a sentence only contains two entities.

To extract both entities and relation from sentence, early works like Zelenko et al. (2003); Chan and Roth (2011) adopted pipeline methods. However, such pipeline methods neglect the relevance between entities and relation. Latter works focused on joint models that extract entities and relation jointly. Yu and Lam (2010); Li and Ji (2014); Miwa and Bansal (2016) relied on NLP tools to do feature engineering, which suffered from the error propagation problem. Miwa and Sasaki (2014); Gupta et al. (2016); Zhang et al. (2017) applied neural networks to jointly extract entities and relations. They converted the relation extraction task into a table filling task. Zheng et al. (2017) took a step further and converted this task into a tagging task. They assigned a semantic tag to each word in the sentence and collected triplets according to the tag information. Bekoulis et al. (2018b,a) model the relation extraction task as a multi-head selec-

tion problem. However, these models can not take triplet's extraction order into consideration. Sun et al. (2018) proposed a joint learning paradigm based on minimum risk training. Their method ignore the influence between relational facts. Zeng et al. (2018b) proposed an sequence-to-sequence model with copy mechanism to handle the overlapping problem in multiple relation extraction. They randomly choose a extraction order for each sentence.

RL has attracted lot of attention recently. It has been successfully applied in many games (Mnih et al., 2015; Silver et al., 2016). Narasimhan et al. (2015); He et al. (2016) applied RL on text based games. Narasimhan et al. (2016) employed deep Q-network to optimize a reward function that reflects the extraction accuracy while penalizing extra effort. Li et al. (2016) applied policy gradient method to model future reward in chatbot dialogue. They designed a reward to promote three conversational properties: informativity, coherence and ease of answering. Su et al. (2016) using on-line activate reward learning for policy optimization in spoken dialogue systems because the user feedback is often unreliable and costly to collect. Yu et al. (2017) applied RL method to overcome the limitations that the Generative Adversarial Net (GAN) in generating sequences of discrete tokens. Our work is related to Li et al. (2016); Yu et al. (2017) since we also apply RL to generate better sequences.

There are several works that related to both relation extraction and RL, which are also related to our work. Zeng et al. (2018a); Feng et al. (2018); Qin et al. (2018) applied RL to distantly supervised relation extraction task. Zeng et al. (2018a) turned the bag relation prediction into an RL process. They assumed that the relation of the bag is determined by the relation of sentences from the bag. They set the final reward to +1 or -1 by comparing the predict bag relation with the gold relation. Feng et al. (2018) adopted policy gradient method to select high-quality sentences from the bag. The selected sentences are feed to the relation classifier and the relation classifier provides rewards to the instance selector. Similarly, Qin et al. (2018) explored a deep RL strategy to generate the false-positive indicator. Our work is different from them since we focus on supervised relation extraction task.

3 Method

We first introduce our basic model and then introduce how to apply RL on it. Similar to Zeng et al. (2018b), our neural model is also a sequence-to-sequence model with copy mechanism. It reads in a raw sentence and generates triplets one by one. Instead of training the model with NLL loss, we regard the triplets generation process as a RL process and optimize the model with REINFORCE (Williams, 1992) algorithm. Therefore, we don't have to determine the triplets order of each sentence beforehand, we let the model generate triplets freely. We show the RL process in Figure 2.

3.1 Sequence-to-Sequence Model with Copy Mechanism

The sequence-to-sequence model with copy mechanism is a kind of CopyNet (Gu et al., 2016) or PointerNetwork (Vinyals et al., 2015). Two components included in this model: encoder and decoder. The encoder is a bi-directional recurrent neural network, which is used to encode a variable-length sentence into a fixed-length vector. We denote the outputs of encoder as $\mathbf{O}^E = [\mathbf{o}_1^E, ..., \mathbf{o}_n^E]$, where $\mathbf{o}_i^E$ denotes the output of i-th word of the encoder and n is the sentence length.

The decoder is another recurrent neural network, which is used to generate triplets one by one. The *NA*-triplets will be generated if the valid triplets number is less than the maximum triplets number.[1] It takes three-time steps to generate one triplet. That is, in time step t ($t = 1, 2, 3, ..., T$), if $t\%3 = 1$, we predict the relation. If $t\%3 = 2$, we copy the first entity and if $t\%3 = 0$, we copy the second entity. T is the maximum decode time step. Note that T is always divisible by 3.

Suppose there are m predefined valid relations, in time step t ($t = 1, 4, 7, ...$), we calculate the confidence score for each valid relation:

$$\mathbf{q}_t^r = selu(\mathbf{o}_t^D \cdot \mathbf{W}_t^r + \mathbf{b}_t^r) \quad (1)$$

where $\mathbf{o}_t^D$ is the output of decoder in time step t; $\mathbf{W}_t^r$ is the weight matrix and $\mathbf{b}_t^r$ is the bias in time step t; $selu(\cdot)$ (Klambauer et al., 2017) is activation function. To allow the model to generate *NA*-triplet, we also calculate the confidence score for

[1] *NA*-triplet is a special triplet proposed in Zeng et al. (2018b). It's similar to the "eos" symbol in neural sentence generation.

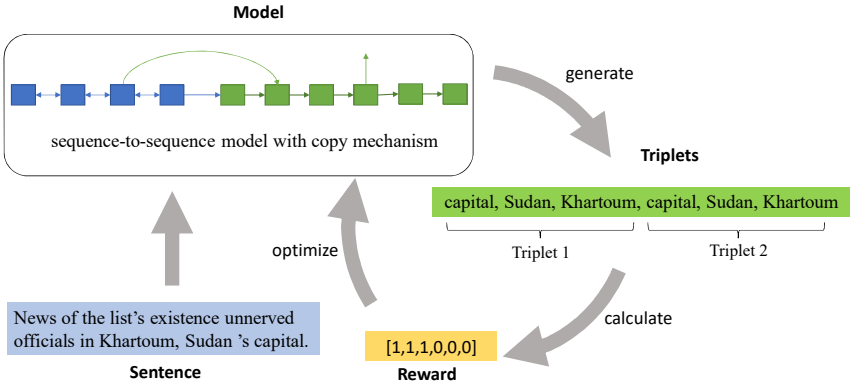

Figure 2: The RL process. The model reads in a raw sentence and generates triplets. Then, a reward is assigned to each time step based on the generated triplets. Lastly, the rewards is used to optimize the model.

NA relation:

$$\mathbf{q}_t^{NA} = selu(\mathbf{o}_t^D \cdot \mathbf{W}_t^{NA} + \mathbf{b}_t^{NA}) \quad (2)$$

where $\mathbf{W}_t^{NA}$ and $\mathbf{b}_t^{NA}$ are parameters in time step t. Then we concatenate $\mathbf{q}_t^r$ and $\mathbf{q}_t^{NA}$ and perform softmax to obtain the probability distribution:

$$\mathbf{p}_t^r = softmax([\mathbf{q}_t^r; \mathbf{q}_t^{NA}]) \quad (3)$$

To copy the first entity in time step t ($t = 2, 5, 8, ...$), we calculate the confidence score of each word in source sentence:

$$q_{ti}^e = selu([\mathbf{o}_t^D; \mathbf{o}_i^E] \cdot \mathbf{w}_t^e) \quad (4)$$

where q_{ti}^e is the confidence score of i-th word and $\mathbf{w}_t^e$ is the weight vector, in time step t. Similarly, to take the *NA*-triplet into consideration, we also calculate the confidence score for *NA* entity with Eq 2. We concatenate them and perform softmax to obtain the probability distribution:

$$\mathbf{p}_t^e = softmax([[q_{t1}^e, ..., q_{tn}^e]; \mathbf{q}_t^{NA}]) \quad (5)$$

Copy the second entity in time step t ($t = 3, 6, 9, ...$) is almost the same as the first entity. The only difference is we also apply the mask (Zeng et al., 2018b) to avoid the copied two entities are the same.

Our model is similar to OneDecoder model and MultiDecoder model in Zeng et al. (2018b). Compared with OneDecoder model, our model using different linear transformation parameters in different decoding time step. Compared with MultiDecoder model, our model using only one decoder cell to decode all triplets. In our model, we didn't using attention mechanism because we found that the attention mechanism makes no difference to the results.

3.2 Reinforcement Learning Process

We regard the triplets generation process as RL process. The loop in Figure 2 represents a RL episode. In each RL episode, the model reads in the raw sentence and generate output sequence. Then we gain triplets from the output sequence and calculate rewards based on them. Finally, we optimize the model with REINFORCE algorithm.

State

We use s_t to denote the state of sentence x in decoding time step t. The state s_t contains the already generated tokens $\hat{y}^{<t}$, the information of source sentence x and the model parameters θ.

$$s_t = (\hat{y}^{<t}, x, \theta) \quad (6)$$

Action

The action is what we predict (or copy) in each time step. In time step t and $t\%3 = 1$, the model (policy) is required to determine the relation of the triplet; In time step t where $t\%3 = 2$ or 0, the model is required to determine the first or second entity, which is copied from the source sentence. Therefore, the action space A is varied in different time step t.

$$A = \begin{cases} R, & t\%3 = 1 \\ P, & t\%3 = 2, 0 \end{cases} \quad (7)$$

where R is the predefined relations and P is the positions of source sentence. We denote the action sequence of the source sentence as $a = [a_1, ..., a_T]$.

Algorithm 1 Reward Assignment

Input: Sampled action sequence $a = [a_1, ..., a_T]$; Gold triplets set G; *NA*-triplet *NA*.
Output: Rewards of each action $r = [r_1, ..., r_T]$.

Number of generated triplets $K = T/3$;
Generated triplets list $F = [F_1, ..., F_K]$;
Already generated triplets set $V = \{\}$.
for each $i \in [1, K]$ **do**
  $F_i = [a_{3*i-2}, a_{3*i-1}, a_{3*i}]$
end for
for each $i \in [1, K]$ **do**
  $r_{3*i-2} = r_{3*i-1} = r_{3*i} = 0$
  if $i \leq |G|$ **then**
    if $F_i \in G$ and $F_i \notin V$ **then**
      Add F_i to V
      $r_{3*i-2} = r_{3*i-1} = r_{3*i} = 1$
    end if
  else if $F_i =$ *NA* **then**
    $r_{3*i-2} = r_{3*i-1} = r_{3*i} = 0.5$
  end if
end for

Reward

The reward is used to guide the training, which is critical to RL training. However, we can't assign a reward to each step directly during the generation since we don't know whether each action we choose is good or not before we finish the generation. Remind that we could obtain a triplet in every three steps. Once we obtained a triplet, we can compare it with the gold triplets and know if this triplet is good or not. A well generated triplet means it's the same with one of the gold triplets and not the same with any already generated triplets.

When we obtained a good triplet after three steps, we assign reward 1 to each of these three steps. Otherwise, we assign reward 0 to them. After generating valid triplets, we may need to generate *NA*-triplets. We assign reward 0.5 to each of these three steps if we correctly generate *NA*-triplet and reward 0 otherwise. We show the details of the reward assignment in Algorithm 1[2].

3.3 Training

The model can be trained with either supervised learning loss or reinforcement learning loss. However, the supervised learning forces the model to generate triplets in the order of the ground truth while the reinforcement learning allows the model generate triplets freely.

NLL Loss

Training the model with NLL loss requires a predefined ground truth sequence for each sentence. Suppose T is the maximum time step of decoder, we denote the ground truth sequence as $[y_1, ..., y_t, ..., y_T]$. Them the NLL loss for sentence x can be defined as:

$$L = \frac{1}{T}\sum_{t=1}^{T} -log(p(y_t|\hat{y}^{<t}, x, \theta)) \quad (8)$$

where $\hat{y}^{<t}$ is the already generated tokens; $p(\cdot|\cdot)$ is the conditional probability; θ is the parameters of the entire model.

RL Loss

Training the model with reinforcement learning only require the ground truth triplets for each sentence. The RL loss for sentence x is:

$$L = \frac{1}{T}\sum_{t=1}^{T} -log(p(\hat{y}_t|\hat{y}^{<t}, x, \theta)) * r_t \quad (9)$$

where $\hat{y}_t$ is the sampled action and r_t is the reward, in time step t.

4 Experiments

4.1 Dataset

Following Zeng et al. (2018b), we test our method on two open datasets: New York Times (NYT) dataset and WebNLG dataset.

NYT dataset is proposed by Riedel et al. (2010). This dataset is produced by distant supervision method which automatically aligns Freebase with New York Times news articles. Like Zheng et al. (2017); Zeng et al. (2018b) do, we ignore the noise in this dataset and use it as a supervised dataset. We use the pre-processed dataset used in Zeng et al. (2018b), which contains 5000 sentences in the test set and 5000 sentences in the validation set and 56195 sentences in the train set. In the train set, there are 36868 sentences that contain one triplet, 19327 sentences that contain multiple triplets. In the test set, the sentence number are 3244 and 1756, respectively. There are 24 relations in total.

WebNLG dataset is proposed by Gardent et al. (2017). This data set is originally created for Natural Language Generation (NLG) task. Given a

[2]How to determine the reward in RL is difficult. We tried several different reward assignments but only this one works.

group of triplets, annotators are asked to write a sentence which contains the information of all triplets in this group. We use the dataset preprocessed by Zeng et al. (2018b) and the train set contains 5019 sentences, the test set contains 703 sentences and the validation set contains 500 sentences. In the train set, there are 1596 sentences that contain one triplet, 3423 sentences contain multiple triplets. In the test set, the sentence number are 266 and 437, respectively. There are 246 different relations.

4.2 Settings

Zeng et al. (2018b) only use LSTM as the model cell. In this paper, we report the results of both LSTM and GRU (Cho et al., 2014). We follow the most settings from Zeng et al. (2018b). The cell unit number is set to 1000; The embedding dimension is set to 100; The batch size is 100; The maximum time step T is 15, that is, we will extract 5 triplets for each sentence; We use Adam (Kingma and Ba, 2015) to optimize parameters and stop the training when we find the best result in validation set. For the NLL training, the learning rate in both dataset is 0.001. For the RL training, we first pretrain the model with NLL training (pretrain model achieves 80%-90% of the best NLL training performance), then training the model with RL. The RL learning rate is 0.0005.

4.3 Evaluation Metrics

We follow the evaluation metrics in Zeng et al. (2018b). Our model can only copy one word for each entity and we use the last word of each entity to represent them. Triplet is regarded as correct when its relation, the first entity and the second entity are all correct. For example, suppose the gold triplets is $< Barack\ Obama, president, USA >$, $< Obama, president, USA >$ is regarded as correct while $< Obama, locate, USA >$ and $< Barack, prsident, USA >$ are not. A triplet is regarded as *NA*-triplet when and only when its relation is NA relation and it has a NA entity pair. The predicted *NA*-triplet will be excluded. We use the standard micro Precision, Recall and F1 score to evaluate the results.

4.4 Results of Different Extraction Order

To find out if the triplets extraction order of a sentence can make difference in multiple relation extraction task, we conduct widely experiments on both NYT and WebNLG dataset. We show the results of different extraction order of different models with LSTM cell in Table 1. The results of models with GRU cell are shown in Appendix B. We box the best results of a model and the bold values are the best results in this dataset.

CNN denotes the baseline with CNN classifier. We use the NLTK toolkit[3] to recognize the entities first. Then we combine every two entities as an entity pair. Every two entities can lead to two different entity pairs. For each entity pair, we apply a CNN classifier (Zeng et al., 2014) to determine the relation. We leave the details of this model in Appendix A. **ONE** and **MULTI** denotes the OneDecoder model and MultiDecoder model in Zeng et al. (2018b).[4] **NLL** means the model is trained with NLL loss, which requires a predefined ground truth sequence for each sentence. For a sentence with N triplets, there are $N!$ (the factorial of N) possible extraction order, which lead to $N!$ valid sequences. **Shuffle** means we randomly select one valid sequence as the ground truth sequence in every training epoch for a sentence. **FixUnsort** means we randomly select one valid sequence before training, and use the selected one as ground truth sequence during training. This strategy is used in Zeng et al. (2018b). **Alphabetical** means we sort the triplets of a sentence in alphabetical order and build ground truth sequence based on the sorted triplets. **Frequency** means we sort the triplets of a sentence based on the relation frequency. We count the relation frequency from the training set. **RL** means the model is trained with reinforcement learning. In NYT dataset, we using Alphabetical strategy to pretrain the model, and in WebNLG dataset, we pretrain the model wtih Frequency strategy.

Form Table 1, we can observe that: (a) The CNN baseline is not performing well because this model neglect the influence between triplets. (b) Compared with FixUnsort strategy, simply change the ground truth sequence in different training epoch (the Shuffle strategy) is also not good. The performance of OneDecoder drops from 0.566 to 0.552 in NYT dataset and 0.305 to 0.283 in WebNLG dataset. (c) In both dataset and for all models trained with NLL loss, sort the triplets in some order (Alphabetical or Frequency order)

[3] https://www.nltk.org/

[4] We reimplement these two models and find that the attention mechanism is not important. Therefore, we report the results without applying the attention mechanism.

Model	NYT			WebNLG		
	Precision	Recall	F1	Precision	Recall	F1
CNN	0.468	0.620	0.533	0.304	0.417	0.352
ONE+NLL (Shuffle)	0.635	0.488	0.552	0.400	0.219	0.283
ONE+NLL (FixUnsort)	0.606	0.530	0.566	0.320	0.291	0.305
ONE+NLL (Alphabetical)	0.616	0.553	0.583	0.305	0.292	0.298
ONE+NLL (Frequency)	0.609	0.546	0.576	0.323	0.311	0.317
ONE+RL	0.715	0.533	0.610	0.668	0.307	0.421
MULTI+NLL (Shuffle)	0.629	0.522	0.570	0.363	0.278	0.315
MULTI+NLL (FixUnsort)	0.600	0.562	0.580	0.434	0.431	0.432
MULTI+NLL (Alphabetical)	0.685	0.647	0.665	0.482	0.481	0.481
MULTI+NLL (Frequency)	0.648	0.604	0.625	0.519	0.518	0.518
MULTI+RL	0.742	0.639	0.687	0.611	0.523	0.564
Our+NLL (Shuffle)	0.670	0.545	0.601	0.417	0.310	0.356
Our+NLL (FixUnsort)	0.645	0.591	0.617	0.490	0.488	0.489
Our+NLL (Alphabetical)	0.717	**0.678**	0.697	0.533	0.537	0.535
Our+NLL (Frequency)	0.688	0.652	0.669	0.581	0.587	0.584
Our+RL	**0.779**	0.672	**0.721**	**0.633**	**0.599**	**0.616**

Table 1: Results of different extraction order of models with LSTM cell.

can lead to better performance. For example, our model achieves 0.617 F1 score under FixUnsort strategy, while 0.697 and 0.669 F1 score under Alphabetical and Frequency strategy in NYT dataset. This observation verifies that the triplets extracting order of a sentence is important in multiple relation extraction task. (d) Another interesting observation is that the NLL trained model can achieve the best F1 score in NYT dataset if we sort the triplets in alphabetical order, while in WebNLG dataset, we need to sort the triplets in relation frequency order. This observation demonstrates that a global sorting rule may not fit for every dataset. The Alphabetical strategy is better for NYT dataset while the Frequency strategy is better for WebNLG dataset. (e) We can also observe that the model trained with RL can achieve better result than any NLL sorting strategy. For example, in WebNLG dataset, MultiDecoder model only achieves 0.481 and 0.518 F1 score with Alphabetical and Frequency strategy, it achieves 0.564 F1 score with RL. Our model trained with RL achieves the best performance on both NYT and WebNLG dataset, which is 0.721 and 0.616. It's 3.4% and 5.5% improvements compared with the best global sorting rule on these two datasets.

4.5 Extraction Order Comparison

This experiment compares the extraction order of our model trained with RL. We test our model on WebNLG dataset. The generated triplets (excluding NA-triplets) sequence of our model which is pretrained with FixUnsort strategy then trained with RL, is denoted as **FURL**. Similarly, the triplets sequence of our model which is pretrained with Frequency strategy then trained with RL is denoted as **FreqRL**. And the triplets sequence of our model which is trained with FixUnsort strategy is denoted as **FUNLL**.

Comparison	LSTM	GRU
FUNLL-FURL	0.326	0.390
FreqRL-FURL	0.446	0.435

Table 2: The extraction order comparison.

Suppose $A = [F_a, F_b, F_c]$ is the generated triplets sequence of FURL for sentence x, $B = [F_a, F_c]$ is the generated triplets sequence of FUNLL for the same sentence x. F_a, as well as F_b and F_c, is a triplet. The first triplet of the sequence A is F_a, which is the same with the first triplet of the sequence B. But the second triplet of A is different from B. Therefore, there are only 1 triplet is in the same position for A and B. The triplets number is the maximum triplets number of A and B, which is 3 in this example. We calculate the order comparison of sentence x as $1/3 = 0.333$. The order comparison of FUNLL and FURL (denote as FUNLL-FURL) is the mean value of all

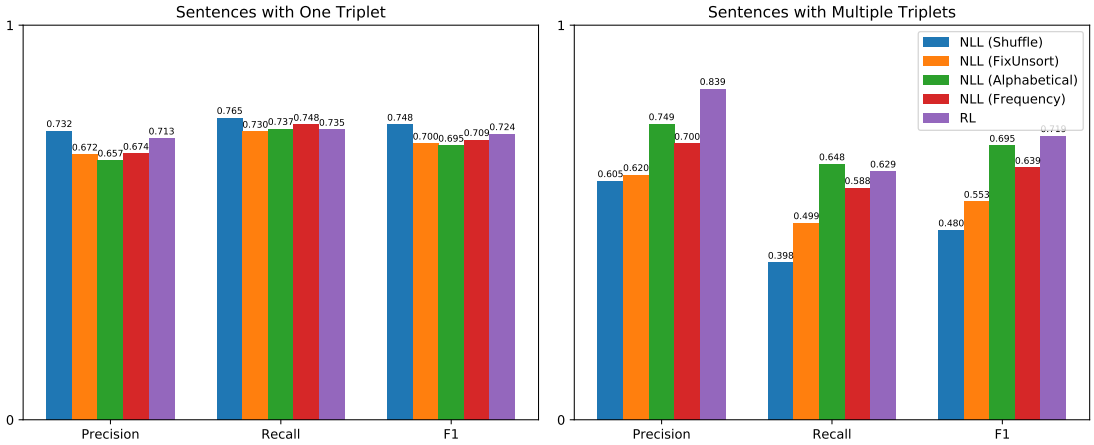

Figure 3: Results of our model with LSTM cell under different training strategies on NYT dataset.

sentences.

We show the order comparison results of our model with LSTM and GRU cell in Table 2. As we can see, although FURL model is pretrained by FUNLL, FURL is more alike FreqRL (0.446 and 0.435), rather then FUNLL (0.326 and 0.390).

This experiment verified that after RL training, the model trend to generate triplets in the same order.

4.6 Multiple Relation Extraction

To verify the ability of extracting multiple relational facts, we conduct the experiment on NYT dataset of our model with LSTM cell. We show the results in Figure 3. The left part of this figure shows the performance of sentences with one triplet. The right part shows the performance of sentences with multiple triplets.

As we can see, when the sentence only contains one triplet, our model trained with RL can achieve comparative performance with the strong baselines. When there are multiple triplets for a sentence, our model trained with RL outperform all baselines significantly. By training with RL, our model could extract triplets more precisely. Although the recall value is slightly lower then NLL training with Frequency strategy, it exceeds other baselines significantly. These observations demonstrate that RL training is effective to handle the multiple relation extraction task.

5 Weakness

Although we overcome all strong baselines by training the model with RL, there are still some weaknesses in our method.

The first weakness is the decrease in recall. Table 1 shows that NLL training with Alphabetical or Frequency strategy achieves the highest recall in most cases. Training the model with RL achieves the highest precision and relatively low recall. This phenomenon demonstrates that the model trained with RL generates relatively fewer triplets. Although we can extract triplets more accurate, it is still a weakness of our method since we try to extract all triplets from a sentence.

The second weakness is our model can only copy one word for each entity. Following Zeng et al. (2018b), we only copy the last word of an entity. But in reality, most entities contains more than one word. In the future, we will consider how to extract the complete entity. For example, we could add the BIO tag prediction in the encoder and train the BIO loss together with current loss. Therefore, we can recognize the complete entity with the help of BIO tags. Or, we can take two steps to generate one entity, one step for the head word and the other for the tail word.

6 Conclusions

In this paper, we discuss the multiple triplets extraction order problem in the multiple relation extraction task. We propose a sequence-to-sequence model with reinforcement learning to take the extraction order into consideration. Widely experiments on NYT dataset and WebNLG dataset are conducted and verified that the proposed method is effective in handling this problem.

7 Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61533018, No.61702512) and the independent research project of National Laboratory of Pattern Recognition.

References

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018a. Adversarial training for multi-context joint entity and relation extraction.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018b. Joint entity recognition and relation extraction as a multi-head selection problem.

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

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the EMNLP*, pages 1724–1734.

Zihang Dai, Lei Li, and Wei Xu. 2016. Cfo: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of ACL*, pages 800–810.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from nosy data. In *Proceedings of AAAI*.

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.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of ACL*, pages 1621–1630.

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.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep Reinforcement Learning for Dialogue Generation. In *Proceedings of EMNLP*, pages 1192–1202.

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.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of EMNLP*, pages 1–11.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of EMNLP*, pages 2355–2365.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of ACL*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML PKDD*, pages 148–163.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL*, pages 626–634.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Pei-Hao Su, Milica Gasic, Nikola Mrki, Lina M. Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of ACL*, pages 2431–2441.

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 EMNLP*, pages 2256–2265.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in NIPS*, pages 2692–2700.

Ronald J Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8:229–256.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceedings of ACL*, pages 118–127.

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.

Wentau Yih, Mingwei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*, pages 1321–1331.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of AAAI*, pages 2852–2858.

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.

Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018a. Large scaled relation extraction with reinforcement learning. In *Proceedings of AAAI*.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018b. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the ACL*.

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.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of ACL*, pages 207–212.

A The Details of the CNN Baseline

In this section, we will describe the details of the CNN baseline. This baseline is a pipeline method. For a sentence, we use the NLTK toolkit to recognize the entities first. Then, we combine each two entities as an entity pair and use a CNN relation classifier to predict their relation.

For example, suppose we recognize 3 entities in sentence s, which denoted as e_1, e_2, e_3. There are 6 different entity pairs (remind that $< e_1, e_2 >$ and $< e_2, e_1 >$ are different).

The CNN classifier is basically the same with Zeng et al. (2014). Each word is turned into a embedding which including it's word embedding and position embedding. After the convolution layer, we apply a maxpooling layer on it. Then we apply a two layer softmax classify layer to obtain the final results. We train the model with NLL loss.

Specifically, the word embedding dimension is 100, the position embedding dimension is 5, we use 128 filters and the filter size is 3. The hidden layer size of softmax classifier is 100 and we use tanh as the activation function. We optimize the model with Adam optimizer (Kingma and Ba, 2015).

During evaluation, if the entity pair is classified into *NA* relation, we will exclude this triplet. Otherwise, the triplet will be regarded as a predict triplet. If the predicted triplet is the same as one of the gold triples, it will be regarded as correct triplet. To be fair, when comparing the entities in the triplets, we only compare the last word of each entity. As long as the last word of the extract entity is the same as the gold one, we regard it as correct.

Model	NYT			WebNLG		
	Precision	Recall	F1	Precision	Recall	F1
ONE+NLL (Shuffle)	0.617	0.471	0.534	0.360	0.214	0.268
ONE+NLL (FixUnsort)	0.597	0.511	0.551	0.311	0.294	0.302
ONE+NLL (Alphabetical)	0.570	0.526	0.547	0.310	0.292	0.300
ONE+NLL (Frequency)	0.609	0.531	0.567	0.311	0.302	0.306
ONE+RL	0.723	0.540	0.618	0.613	0.300	0.403
MULTI+NLL (Shuffle)	0.617	0.517	0.562	0.377	0.275	0.318
MULTI+NLL (FixUnsort)	0.586	0.544	0.564	0.404	0.403	0.403
MULTI+NLL (Alphabetical)	0.652	0.635	0.643	0.470	0.471	0.471
MULTI+NLL (Frequency)	0.636	0.593	0.614	0.500	0.499	0.500
MULTI+RL	0.738	0.636	0.683	0.615	0.478	0.538
OUR+NLL (Shuffle)	0.644	0.545	0.591	0.419	0.312	0.358
OUR+NLL (FixUnsort)	0.657	0.577	0.615	0.488	0.485	0.486
OUR+NLL (Alphabetical)	0.701	**0.683**	0.692	0.521	0.516	0.518
OUR+NLL (Frequency)	0.680	0.642	0.660	0.569	**0.576**	0.572
OUR+RL	**0.770**	0.671	**0.717**	**0.628**	0.555	**0.589**

Table 3: Results of different extraction order of models with GRU cell.

B Results of GRU Cell

We show the results of different extraction order of models with GRU cell in Table 3.