applied sciences

MDPI

Article

A Text-Generated Method to Joint Extraction of Entities and Relations

Haihong E *, Siqi Xiao and Meina Song

School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China; xiaosiqi@bupt.edu.cn (S.X.); mnsong@gmail.com (M.S.)

* Correspondence: ehaihong@bupt.edu.cn; Tel.: +86-156-0010-5933

Received: 26 July 2019; Accepted: 2 September 2019; Published: 10 September 2019

Abstract: Entity-relation extraction is a basic task in natural language processing, and recently, the use of deep-learning methods, especially the Long Short-Term Memory (LSTM) network, has achieved remarkable performance. However, most of the existing entity-relation extraction methods cannot solve the overlapped multi-relation extraction problem, which means one or two entities are shared among multiple relational triples contained in a sentence. In this paper, we propose a text-generated method to solve the overlapped problem of entity-relation extraction. Based on this, (1) the entities and their corresponding relations are jointly generated as target texts without any additional feature engineering; (2) the model directly generates the relational triples using a unified decoding process, and entities can be repeatedly presented in multiple triples to solve the overlapped-relation problem. We conduct experiments on two public datasets—NYT10 and NYT11. The experimental results show that our proposed method outperforms the existing work, and achieves the best results.

Keywords: relation extraction; entity recognition; information extraction; long short-term memory network

1. Introduction

Entity-relation extraction is the core task and important segment in the fields of information extraction, knowledge graph, natural language understanding, etc. In recent years, knowledge graph [1] has been widely applied. Many achievements have also been made in the downstream tasks such as question answering and retrieval based on knowledge graph. The basis for constructing the knowledge graph is to build a knowledge base. In the knowledge base, the structured relational triples are preserved in formats such as <*entity 1*, *rel*, *entity 2*>, which means that there is a relation *rel* between *entity 1* and *entity 2*. The goal of entity-relation extraction task is to extract the semantic relations between entity pairs from unstructured text. With the application of deep learning in joint learning and distant supervision, the relation extraction task has obtained rich research results.

The supervised entity-relation extraction methods can be divided into pipeline and joint learning. The pipeline methods take the named entity recognition (NER) and relation classification (RC) as two separate subtasks, and extract the relations between entities based on the completion of entity recognition [2–4]. However, this kind of methods ignores the relevance between these two subtasks. Recent work using joint learning [5–8] can make use of the tight interaction information between entities and relations and use a single model to extract entities and classify the relations between entities simultaneously, which solves the problems of the pipeline method well. However, most of the existing work often requires complex feature engineering or relies heavily on the NLP tools to extract features.

Moreover, most existing relation extraction models focus on scenarios dealing with a single relation within one sentence, but there are usually multiple relations between entity mentions in one

sentence. Sentences can be divided into three classes based on the degree of entity overlap [9], as shown in Table 1: (1) Normal: a sentence belongs to Normal class if none of its triplets have overlapped entities; (2) Entity Pair Overlap: some of its triplets have overlapped entity pair; (3) Single Entity Overlap: some of its triplets have an overlapped entity and these triplets do not have overlapped entity pair. Even though there are already several works to address the triplet overlap issue [9–11], their effects are far from good enough, cannot solve the problems of relation extraction in complex situations very well. As a result, the triplet overlap issue is not actually addressed.

Table 1. Examples of three classes: Normal, Entity Pair Overlap, and Single Entity Overlap. S1 belongs to Normal class because there are no overlaps in its triplets; S2 belongs to Entity Pair Overlap class since the entity pair (Sudan, Khartoum) are overlapped; S3 belongs to Single Entity Overlap class because the entity *Los Angeles* is overlapped and its two triplets have no overlapped entity pairs.

Class	Sentence	Relation Triples
Normal	S1: Chicago is in the United States.	<The United States, **contains**, Chicago>
Entity Pair Overlap	S2: News of the list's existence unnerved officials in Khartoum, Sudan's capital.	<Sudan, **contains**, Khartoum> <Sudan, **capital**, Khartoum>
Single Entity Overlap	S3: John, 23, who lives in Los Angeles, California.	<John, **placelived**, Los Angeles> <California, **contains**, Los Angeles>

In reality, natural language texts, such as news and blogs, usually express multiple relations and it is also common that one or more entity mentions appear among multiple relations. Therefore, it is necessary to extract overlapping relations from the perspective of practical application scenarios. The overlapping multi-relation extraction problem is more complex than single-relation extraction because the single-relation extraction scenario can be basically divided into the following two types: (1) for sentences with the given entity pairs, relation classification can be modeled as a text classification problem; (2) for sentences with non-annotated target entity pairs, the model assumes that sentence contains only one pair of entities and relation classification is performed after entity recognition; these two cases usually use the *softmax* function in the relation classification phrase, so only one relation can be extracted. In the multi-relation extraction situation, we need to find every complete relational triple. The model needs to simultaneously extract the relation and the corresponding entity pairs. Reference [8] provided an idea to integrate entity mention and relation type information into each label, this two information can be obtained simultaneously when tagging each word. Zeng's [9] work is similar to ours, which is based on a sequence-to-sequence learning framework, but it cannot extract multi-word entities because of the model design.

To tackle this problem, we completely convert entity-relation extraction task into text generation task. We generate entity pairs and relational representation words according to source texts, without any additional feature engineering. The task of generating target texts from source texts, including text summarization [12,13], machine translation [14,15]. In the text summarization task, target texts are keywords or key sentences that are copied from source texts or generated from vocabulary through the semantic understanding of source text contents. For the relation extraction task, our target texts are the entity pairs contained in the source texts and their corresponding relations, i.e., relational triplets.

In this paper, we adopt a sequence-to-sequence framework with the pointer, where using the encoder to obtain the semantic encoding vector and the decoder with pointer is used to generate entities or relations. Inspired by the text summarization paper [16], we also use a generation probability p_{gen} as a soft switch to select whether the current decoding time is more likely to copy words from the original input or to generate words from the vocabulary. According to the specific situation of the original input, one or more groups of relational triplets are generated, thereby implementing the joint extraction of entities and relations. Entities can be repeated in multiple triplets, which can solve the problem of overlapped multiple relational triplet extraction.

The main contributions of our work are concluded as follows:

(1) We completely convert the entity-relation extraction to the text generation task, and use a unified decoding method to generate entities and relational expressions as target text to realize the joint extraction of entities and relations.
(2) Based on the text generation framework, the model is designed to generate multiple groups of relational triplets. Entities can be repeated in multiple triplets to solve the problem of overlapped multiple relational tuples.
(3) We conduct experiments on NYT10 and NYT11 public datasets, and the experimental results show that we proposed method outperforms state-of-the-art with 4.7% and 11.4% improvements in F1 score, respectively.

The remainder of the paper is organized as follows: Section 2 reviews the related works. Section 3 describes the proposed method in detail. In Section 4, datasets and settings used in the experiment are presented and Section 5 shows the results. Section 6 discusses the performance comparison between our model and the baseline methods. Section 7 concludes the paper.

2. Related Work

Entity and relation extraction methods can be divided into pipeline and joint learning.

Pipeline method regards entity recognition and relation extraction as two separate tasks, and extracts relations based on entity recognition. Some pipeline methods based on RNN and CNN models have been proposed. Ref. [17] first used RNN for relation extraction. Ref. [4] first introduced CNN to this task. Refs. [2,18–20], improved on the previous work and enhanced the effect of relation extraction. The pipeline method has the disadvantages of error propagation, ignoring the relevance between these two subtasks and generating redundant information, while joint learning method proposed in recent years which uses a single model to extract entities and relations simultaneously and can use the close interaction information between entities and relations.

The joint learning method is further divided into feature-based structured systems [21] and neural network models. Ref. [7] first used neural network methods with the dependency tree to jointly extract entities and relations. Ref. [22] proposed a hybrid neural network which has two channels after the encoding layer, one links to the NER module, the other feeds into the relation extraction module. Ref. [5] first introduced the attention mechanism in combination with bidirectional LSTMs for joint extraction of entities and relations. Ref. [8] proposed an entity-relation extraction method based on a novel tagging scheme. This method completely transforms the joint learning model into sequence labeling problem, it can extract multiple relations, but cannot deal with entity overlap because the model can only assign a label to each word. Ref. [11] based on Zheng's work [8], further transformed the joint task into a graph problem and proposed a transition-based method, can model underlying dependencies between relations and identify overlapped relational triples.

Ref. [9] first proposed a solution for overlapping relation extraction, and divided the sentences into three classes according to the degree of entity overlap: Normal, Entity Pair Overlap, and Single Entity Overlap. They proposed an end-to-end model based on sequence-to-sequence learning with copy mechanism, copying entity pairs from the original input, and classifying the relation types in the predefined relational table. Unlike [9], our model uses a unified way to generate token at any time in the decoding process, instead of judging whether to copy entities or predict relations at different steps. By calculating a generate probability distribution, the model can automatically learn whether entities or relations should be generated at each moment. At the same time, because [9] presupposes that a relational triple is generated every three steps, they can only recognize the last word of a multi-word entity. While our model does not limit the number of words contained in each relational triple, it can recognize multi-word entities and can copy entity words from the original text.

3. Materials and Methods

In this section, we first formalize the description of the entity-relation problem. Then, we introduce the sequence-to-sequence model with the pointer we use in detail.

3.1. Problem Formulation

Giving the training data [x, y], x represents the input text of the model and y represents the target output. In the target sentence, we use '.' to divide multiple triples, and within the triple, we use ',' to divide relational words, the first entity, and the second entity.

The goal of the model is to generate one or more groups of relational triplets according to the specific situation in the source text, while allowing entities to be repeatedly presented. The model can copy words from the source text by the pointer or generate words from the predefined vocabulary. The overall structure of our model is shown in Figure 1.

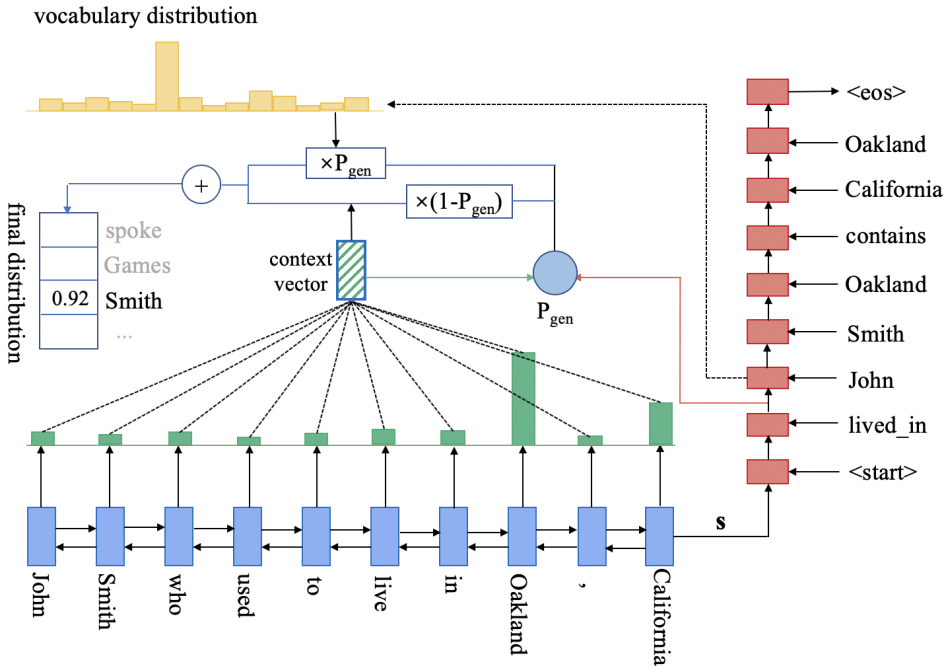

Figure 1. The overall structure of our model. The blue block represents the bidirectional LSTMs in the encoder, the red block represents the unidirectional LSTM in the decoder, the green block represents the attention weight distribution, and the yellow block represents the final generation probability distribution. All these above descriptions will be introduced in Section 3.

3.2. Model Description

3.2.1. Encoder

Giving a sentence s = [w_1, w_2, ... , w_n], where w_t represent the t-th word in the sentence of length n, we first convert the word with one hot encoding into the embedding matrix through the word embedding layer, and get e = [x_1, x_2, ... , x_n], where $x_t \in R^d$ represents the embedding vector of t-th word. The embedding layer randomly initializes the embedding matrix and updates the weight parameters with the training of the model.

Then, we use LSTM to further encode the sequence. Long Short-Term Memory (LSTM) is a variant of Recurrent Neural Network (RNN) which is widely used in various NLP tasks because it has ability to capture long-term dependencies and solve the problem of gradient vanish in RNN. Specifically, we use bidirectional LSTMs (Bi-LSTMS) which consists of two separate LSTM layers. The forward LSTM layer $\overrightarrow{h}$ encodes the input sequence from x_1 to x_n. Similarly, the backward LSTM layer $\overleftarrow{h}$ will

encode the input sequence from x_n to x_1. We then concatenate $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ to represent final encoder information of t-th word, denoted as $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$, in this way, the encoder vector of each step can obtain the semantic information of its context.

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{1}$$

In LSTM, as shown in Equation (1), x_t represents the word vector of the t-th word, and h_t represents the hidden state vector at the time t. W and b represent the weight matrices and bias vectors that can be learned, respectively.

3.2.2. Decoder

The decoder is aimed to generate tokens consisting of a layer of unidirectional LSTM.

$$o_t, s_t = cell([x_{t-1}, h_t^*], s_{t-1}) \tag{2}$$

In Equation (2), *cell* is an LSTM unit, during training x_{t-1} represents the embedding of the previous word in the target output sequence; in the test phase, it represents the embedding of the word generated by the model at the previous step, and s_{t-1} represents the decoding state at time $t-1$. At the same time, we use the attention mechanism to calculate the weight of the hidden vectors in the encoder at the current decoding time, the attention distribution can be viewed as a probability distribution over the source words. The greater the attention weight, the greater the influence on the word generated at the current decoding time. In addition, h_t^* represents the weighted sum of the encoder hidden states based on attention weight, i.e., context vector. We use the attention calculation method of [14] to obtain the context vector:

$$\begin{aligned} e_i^t &= v^T tanh(W_h h_i + W_s s_t + b_{attn}) \\ a^t &= softmax(e^t) \\ h_t^* &= \sum_i a_i^t h_i \end{aligned} \tag{3}$$

where v, W_h, W_s, and b_{attn} in Equation (3) are learnable parameters, h_i represents the hidden state vector of the encoder at time i .

a^t is the influence weight in attention which is also a probability distribution. When the model wants to 'copy' a word from the original text, the word with the largest weight value will be selected as the predicted word. Therefore, we also call a^t 'pointer'.

Then the context vector is concatenated with the decoder state s_t and fed through linear layers to produce the vocabulary distribution P_{vocab}:

$$P_{vocab} = softmax(W_v[s_t, h_t^*] + b_v) \tag{4}$$

where W_v and b_v are learnable parameters, P_{vocab} is a probability distribution over all words in the predefined vocabulary.

To make the model have the ability to copy words from the source text, and to retain the ability to select words through the predefined vocabulary, we calculate a generation probability $p_{gen} \in [0, 1]$ at each decoding step, refer to [16]:

$$p_{gen} = \sigma(w_h^T * h_t^* + w_s^T s_t + w_x^T x_t + b_{gen}) \tag{5}$$

where w_h, w_s, w_x and b_{gen} are learnable parameters and σ is the sigmoid function, p_{gen} is calculated by context vector h_t^*, decoder state s_t, and decoder input x_t. p_{gen} is aimed to select the word output at the current decoding time, with a greater probability of copying from the source text or more likely to be generated from the predefined vocabulary.

And now we get the final probability distribution:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_i a_i^t \tag{6}$$

where a^t represents the attention weight on the hidden states of the encoder. We select the word with the greatest probability as the predicted word of the current step. In the test phase, the embedding of this word will be sent to the next decoding step.

For entities, the model will tend to copy from the source text, so that for entities that do not appear in the predefined vocabulary (unseen entities), the model also has the ability to correctly identify; for relational expressions, the model is more tend to generate from the vocabulary.

3.2.3. Training and Decoding

During training, given a batch of data with B sentences S = $\{s_1, s_2, \ldots, s_b\}$ with their corresponding target sequences Y = $\{y_1, y_2, \ldots, y_b\}$, where $y_i = \{w_i^1, w_i^2, \ldots, w_i^T\}$ is the reference of i-th sentence. The loss function is defined as follows:

$$loss = \frac{1}{B \times T} \sum_{i=1}^{B} \sum_{t=1}^{T} -log(P(w)) \tag{7}$$

where T is the maximum time step of decoder.

While decoding, the model adopts beam search to increase the accuracy of the output. The advantage of beam search is that we have multiple choices at each step, instead of selecting the word with the highest probability at each time, in case that the optimal local prediction is incorrect. The candidate predictions are ranked by global scores; thus, error propagation can be alleviated.

4. Experimental Setup

In this section, we present our experimental results on two different public datasets NYT10 and NYT11, and compare them with the baseline methods to demonstrate the effectiveness of our model from multiple perspectives.

4.1. Dataset

We conduct experiments on two public datasets NYT10 and NYT11. NYT (New York Times) dataset is developed by distant supervision method. The original corpus in this dataset is extracted from sentences in New York Times articles. NYT10 and NYT11 are two versions of NYT dataset. Specifically, NYT10 dataset contains 29 valid relations, including 74,345 sentences, which is originally released by [23]. NYT11 is relatively small, including 24 valid relations, which is provided by [24]. We filtered the sentences that do not contain valid triples in the dataset, leaving 66,336 sentences. The training set, valid set, and test set are split by random sampling. Some statistical data of the two datasets are shown in Table 2.

Table 2. Statistics of NYT10 and NYT11 datasets.

	NYT10	NYT11
Relation types	29	24
Training set	66,828	58,356
Training tuples	84,166	98,393
Test set	4000	4998
Test tuples	5010	8226

4.2. Settings

We set 256 as the hidden state dimension of LSTM, 128 as the word embedding dimension, and the batch size is 16. We set the maximum number of decoding steps to 60, so the model can generate up to 10 groups of relational triples. We use Adam to optimize parameters and learning rate is set to 0.001 during training. We set beam size is 4, which means that the top 4 optimal generated sequences are preserved during the decoding phase, and finally the one with the highest probability is selected as the final output.

4.3. Baseline and Evaluation Metrics

We select four models as our baselines, CoType is a joint extraction model based on feature system. SPTree uses neural network model with abundant linguistic resources. Noveltagging and MultiDecoder both use neural network to jointly extract entities and relations without additional features. These models all achieved the best results at that time.

- CoType [24]: a domain-independent framework by jointly embedding entity mentions, relation mentions, text features, and type labels into representations, which formulates extraction as a global embedding problem.
- SPTree [7]: an end-to-end relation extraction model that represents both word sequence and dependency tree structures using bidirectional sequential and tree-structured LSTM-RNNs.
- Noveltagging [8]: an approach that treats joint extraction as a sequential labeling problem using a tagging schema where each tag encodes entity mentions and relation types at the same time to achieve joint extraction of entities and relations.
- MultiDecoder [9]: a sequence-to-sequence learning framework with a copy mechanism for joint extraction, where multiple decoders are applied to generate triples to handle overlapping relations, completing the extraction of a relational triple every three steps. This method is the first time to solve the overlapping problem of multi-relational extraction.

We compare our method with the above four baselines on NYT10 and NYT11 dataset respectively. In addition, we evaluate the performance of each model with micro Precision, Recall, and F1 score. Only when the relation and entity pair are all correct, we think this relational triplet is correctly predicted, where an entity is considered correct if the head and tail offsets are both correct. We used the source code provided by above baselines to reproduce their performance on NYT10 and NYT11 dataset, respectively.

5. Results

In this section, we will show the experimental results of our proposed method and baseline methods on NYT10 and NYT11, we reproduce the results of the baseline methods.

Model Performance

In which Table 3 shows the comparison of extraction effects on test sets of NYT10 and NYT11, respectively. It can be found that our proposed method is better than baseline methods both on NYT10 and NYT11 datasets, and outperforms [9] with 4.7% and 11.4% improvements in F1 score,

respectively. At the same time, to prove the ability of our model to extract overlapped multi-relations, we respectively divide two subsets from the NYT10 and NYT11 test sets. All sentences in one subset have entity pair overlap, and all sentences in the other subset have single entity overlap. Please note that some sentences may exist in both cases, in this experiment, this kind of sentence will exist in both subsets. We compare our model with Noveltagging and MultiDecoder. Figures 2 and 3 show the precision, recall, and F1 score of entity pair overlap and single entity overlap on NYT10 and NYT11 datasets, respectively. In the figure, blue, yellow, and green blocks represent the experimental results of Noveltagging, MultiDecoder, and our model, respectively. As we can see, our model can handle overlapped multi-relation extraction better than the baseline methods on both datasets.

Table 3. Comparison of results of our model and baselines in NYT10 and NYT11 datasets.

Model	NYT10			NYT11		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
CoType	-	-	-	0.417	0.320	0.362
SPTree	0.464	0.591	0.519	0.493	0.634	0.555
Noveltagging	0.563	0.334	0.419	0.622	0.341	0.440
MultiDecoder	0.543	0.530	0.536	0.586	0.574	0.580
Our Method	**0.592**	**0.533**	**0.561**	**0.702**	**0.598**	**0.646**

Bold numbers represent the results of proposed method and are also the highest scores of the three evaluation metrics (precision, recall and F1 score) in the comparative experiment.

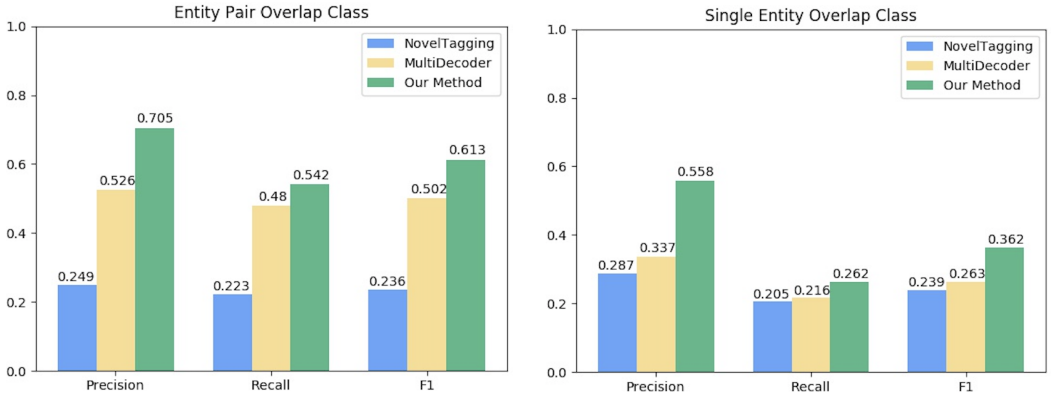

Figure 2. Results of our model and baseline models in Entity Pair Overlap class and Single Entity Overlap class in NYT10 dataset.

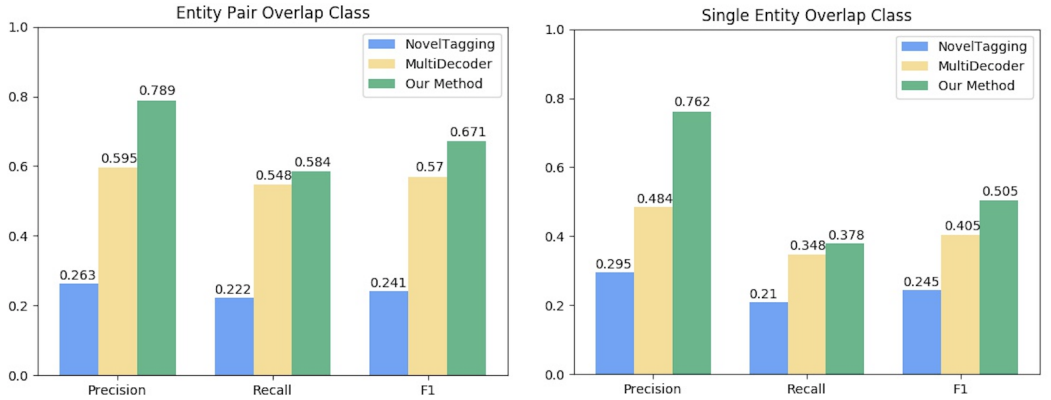

Figure 3. Results of our model and baseline models in Entity Pair Overlap class and Single Entity Overlap class in NYT11 dataset.

6. Discussion

In this section, we focus on the advantages of our model over other baselines, and explain and analyze the experimental results in detail.

6.1. Comparison of Overall Performance

Table 3 shows the Precision, Recall, and F1 scores of the baseline models and our proposed method. CoType is a feature-based system whose performance is not as good as neural network models. SPTree uses more linguistic resources (e.g., POS tags, chunks, syntactic parsing trees) to obtain better results Noveltagging method [8] cannot solve the problem of overlapping multi-relations because it can only assign a tag to each word in the sequence, which leads to a decrease in accuracy. Ref. [9] decides to copy entities or predict relations according to different decoding steps. The copy mechanism is used to calculate the probability distribution to select entities on the source texts at the time of entity recognition, and at the steps of relation prediction, the probability distribution is calculated on the relational table. While, during decoding, we do not distinguish the generation time of entities or relations, and relational words are also distributed in the predefined vocabulary, rather than having a separate relational table. We adopt a more unified decoding method, the predicted word at any time is generated by calculating the mixed probability distribution $P(w)$ over the vocabulary at each decoding step. We hope that the model can learn whether to generate entities or relational words at each step in the process of training.

Meanwhile, we set a maximum decoding step of 60 to generate up to 10 relational triples, while [9] can generate up to 5 relational triples. At the same time, Zeng' s model can extract multiple triples, but it is limited to the $3t + 1$, $3t + 2$ ($5 > t >= 0$) to generate the first entity and the second entity of the current triple. According to its presupposition, multi-word entity cannot be extracted completely, which is a disadvantage in its model design. Our method can extract the whole part of each entity completely, so when we judge whether the model extracts a triple correctly, Zeng' s model is more relaxed than our model, because it is equivalent to just extracting the last word in the entity as if the entity was correctly extracted.

6.2. Comparison of Overlapped Multi-Relations Extraction Performance

To further contrast with baselines, we experiment with sentences of different entity overlap degrees, respectively. Figures 2 and 3 show the experimental results of our proposed method and two of our baseline methods (Noveltagging and MultiDecoder), respectively. As we can see in Entity Pair Overlap class and Single Entity Overlap class, our method performs much better than others. We think that our method generates entities and their relations as target texts, if there are multiple relations between the entity pairs or an entity belongs to multiple triplets, then it can be understood that this entity or entity pair has more abundant semantic information, and these entities will get more attention at the moment of decoding. Therefore, there are greater probabilities for the model to select them from the source texts. Thus, our method is more suitable for processing the relation extraction in entity overlap case than [9]. Again, Noveltagging [8] cannot assign multiple tags to a single word, which makes it impossible to extract overlapped relational triples.

6.3. Comparison of the Multiple Relational Triples Extraction Performance

We further divide the NYT11 test set and classify test set into 7 subclasses according to the relation number of the entity pairs in each sentence. We test the extraction capability of our model and MultiDecoder on each class which contains 1, 2, 3, 4, 5, 6 and >= 7 relational triples, respectively. The results are shown in Figure 4, we can see that as the number of relations contained in a sentence increases, the performance of MultiDecoder decreases. However, when the sentence has one to four relational triples, the effect of our model is gradually increasing, and achieves the best performance

when the number of relations is 4. When the number of relations between the same entity pair is greater than 4, the extraction effects will gradually decrease.

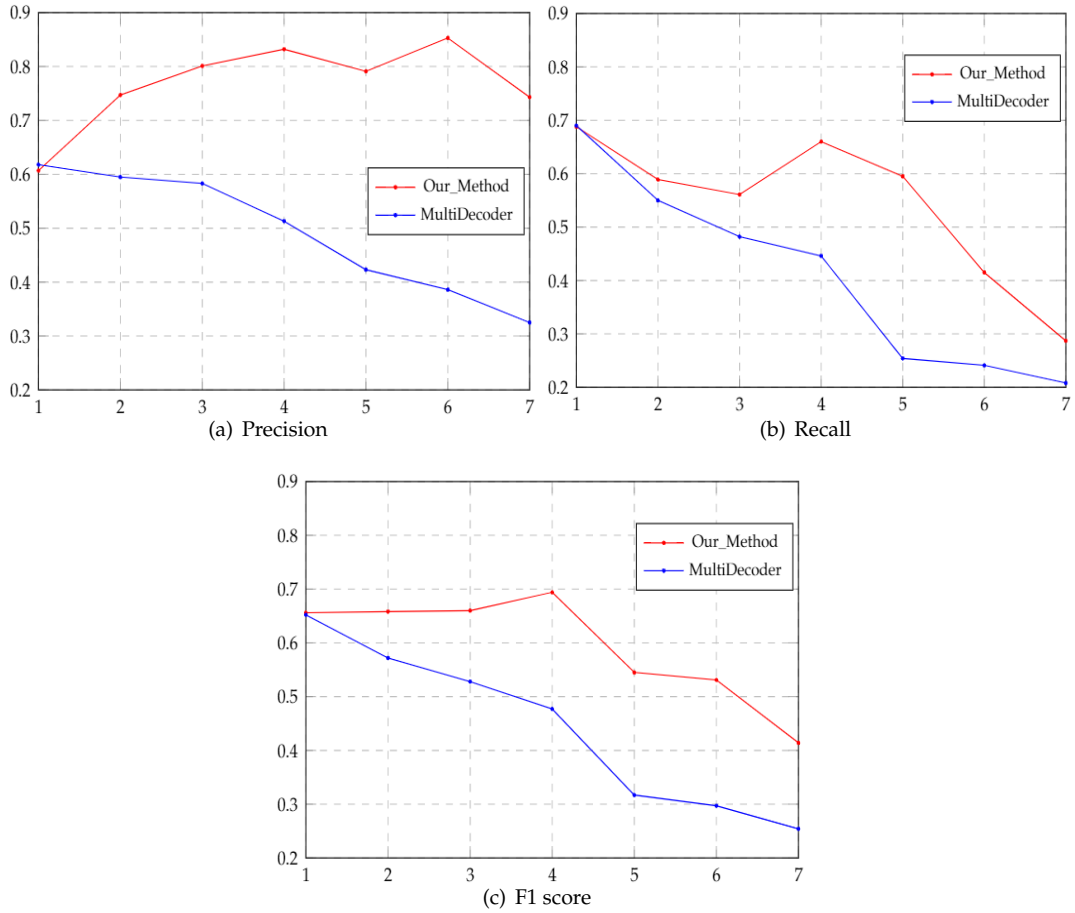

Figure 4. Results of Precision, Recall and F1 score of sentences that contains different number of triplets of our model and MultiDecoder on NYT11 dataset.

As the number of relations increases, extraction becomes more difficult, so the performance of MultiDecoder decreases gradually. For our model, as mentioned earlier, if there are multiple relations between entities, it will be more likely to be noticed when decoding and thus extracted, but this is within a certain threshold range (<=4), when the number of relations continues to increase, our model will also have a performance degradation.

We will analyze why F1 score is the highest when there are 4 relational triples in a sentence from the perspective of the proportion of entity pair overlap in each subclass. From Figures 2 and 3, we can see that our model is more suitable for dealing with entity pair overlap class than single entity overlap class. If there are more than two relations in the sentence, there will usually be one of two types of overlapping situations. Therefore, we analyze the proportion of entity pair overlap situation in these sentences which contain more than two relations. We count the number of relations that an entity pair contains when there are 2, 3, and 4 relations in the sentence respectively, if the entity pair contains 2 or more relations, it means there is entity pair overlap.

Table 4 shows the statistical results. From the table, we can see that in the subclass containing 4 triples in the sentence, has the largest proportion of entity overlap, reaching 80%, which is beyond the other subclasses, so we think this is the reason makes the model perform best on this subclass. The more relations are contained in sentences, the more complex the extraction is. When sentences contain more relations (>4), we consider the following two reasons leading to the performance degradation of the model. First, since our model generates relatively independent words rather than sentences with

contextual contexts, it is relatively weak for LSTM to generate such a long sequence without coherent semantics. Secondly, because the training set contains less than 3% of the sentences with more than 4 relations, the model is not sufficient to learn this situation.

Table 4. Statistics Results.

	1	2	3	>=4	Percentage
sentences containing **2** relation triples	514	943	-	-	0.647
sentences containing **3** relation triples	210	72	182	-	0.547
sentences containing **4** relation triples	70	71	12	194	**0.800**

Bold numbers represent the largest proportion of entity pair overlap in the three cases.

6.4. Case Study

Table 5 shows three examples of our model extracted from the NYT11 dataset, corresponding to three categories: normal, single entity overlap, and entity pair overlap. The first sentence belongs to the normal class and does not have multiple relations. '*contains*' means the relation of entity *America* and entity *Houston*. Our model generate '*contains*' from predefined vocabulary and copy *America*, *Houston* from input text. The second sentence contains two relations in which there is overlap of a single entity, *Italy*. The third sentence contains two relations where entity pairs overlapped, <*Microsoft*, *Bill Gates*>. The entity pair in the last sentence is <*Somerset County*, *Quecreek*>, but the model only copies the last word 'County' in *Somerset County* from the original text, and does not extract entity completely. In this case, we think that the triple predicted by the model is wrong.

Table 5. Extraction examples of our models. The first column in the table is the input of the model, and the second column is the corresponding sentences of the output of the model. As described in Section 3.1 above, multiple relational triples in the model output are separated by '.' and ',' separates relational words and two entities within each triple.

Input	Output of Our Model
Kevin Steurer is helping complete arrangements for a family trip to **Houston** , **America** .	contains , America , Houston .
You can take the train from many cities in **Italy** to **Lecce** , which is about 45 min from **Otranto** by car.	contains , Italy , Lecce . contains , Italy , Otranto .
The real power at **Microsoft** resides with its longtime leaders—**Bill Gates**, the co-founder and chairman.	work_in , Bill Gates , Microsoft . founder , Microsoft , Bill Gates .
Somerset County has experienced disaster , with the crash of flight and nine coal miners trapped at **Quecreek**.	contains, County, Quecreek

7. Conclusions

In this paper, we propose to completely transform the entity-relation extraction task into the text generation task to solve the entity overlap problem in relation extraction. We use a pointer-based sequence-to-sequence framework to enable the model to copy words from the source text or to select words from the predefined vocabulary. We further analyze the extraction ability of our model on different degrees of entity overlap, and classify the sentences according to the different number of relations between two entities, and test the extraction effects of our model on these subclasses. We conduct experiments on the public datasets NYT10 and NYT11. The experimental results show that our method outperforms the baselines.

Author Contributions: Conceptualization, H.E. and S.X.; Data curation, H.E.; Formal analysis, H.E.; Funding acquisition, H.E.; Investigation, S.X.; Methodology, S.X.; Project administration, H.E.; Resources, H.E.; Software, H.E.; Supervision, H.E.; Validation, H.E., S.X. and M.S.; Visualization, H.E.; Writing—original draft, S.X.; Writing—review & editing, H.E., S.X. and M.S.

Funding: This work was supported in part by the National Key R&D Program of China under Grant SQ2018YFB140079.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RC	Relation Classification
NER	Named Entity Recognition
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network

References

1. Zhao, M.; Wang, H.; Guo, J.; Liu, D.; Xie, C.; Liu, Q.; Cheng, Z. Construction of an Industrial Knowledge Graph for Unstructured Chinese Text Learning. *Appl. Sci.* **2019**, *9*, 2720. [CrossRef]
2. Cai, R.; Zhang, X.; Wang, H. Bidirectional recurrent convolutional neural network for relation classification. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany, 7–12 August 2016; Volume 1, pp. 756–765.
3. Hashimoto, K.; Miwa, M.; Tsuruoka, Y.; Chikayama, T. Simple customization of recursive neural networks for semantic relation classification. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 1372–1376.
4. Zeng, D.; Liu, K.; Lai, S.; Zhou, G.; Zhao, J. Relation classification via convolutional deep neural network. In Proceedings of the COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, Dublin, Ireland, 23–29 August 2014; pp. 2335–2344.
5. Katiyar, A.; Cardie, C. 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), Berlin, Germany, 7–12 August 2016; Volume 1, pp. 919–929.
6. Katiyar, A.; Cardie, C. 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), Vancouver, BC, Canada, 30 July–4 August 2017; Volume 1, pp. 917–928.
7. Miwa, M.; Bansal, M. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv* **2016**, arXiv:1601.00770.
8. Zheng, S.; Wang, F.; Bao, H.; Hao, Y.; Zhou, P.; Xu, B. Joint extraction of entities and relations based on a novel tagging scheme. *arXiv* **2017**, arXiv:1706.05075.
9. Zeng, X.; Zeng, D.; He, S.; Liu, K.; Zhao, J. Extracting Relational Facts by an End-to-End Neural Model with Copy Mechanism. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia, 15–20 July 2018; Volume 1, pp. 506–514.
10. Christopoulou, F.; Miwa, M.; Ananiadou, S. 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), Melbourne, Australia, 15–20 July 2018; Volume 2, pp. 81–88.
11. Wang, S.; Zhang, Y.; Che, W.; Liu, T. Joint Extraction of Entities and Relations Based on a Novel Graph Scheme. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; pp. 4461–4467.
12. Nallapati, R.; Zhai, F.; Zhou, B. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
13. Zhang, Y.; Li, D.; Wang, Y.; Fang, Y.; Xiao, W. Abstract Text Summarization with a Convolutional Seq2seq Model. *Appl. Sci.* **2019**, *9*, 1665. [CrossRef]
14. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
15. Cheng, Y.; Yang, Q.; Liu, Y.; Sun, M.; Xu, W. Joint training for pivot-based neural machine translation. In Proceedings of the IJCAI, Melbourne, Australia, 19–25 August 2017.

16. See, A.; Liu, P.J.; Manning, C.D. Get to the point: Summarization with pointer-generator networks. *arXiv* **2017**, arXiv:1704.04368.
17. Socher, R.; Huval, B.; Manning, C.D.; Ng, A.Y. Semantic compositionality through recursive matrix-vector spaces. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Korea, 12–14 July 2012; pp. 1201–1211.
18. Xu, K.; Feng, Y.; Huang, S.; Zhao, D. Semantic relation classification via convolutional neural networks with simple negative sampling. *arXiv* **2015**, arXiv:1506.07650.
19. Santos, C.N.D.; Xiang, B.; Zhou, B. Classifying relations by ranking with convolutional neural networks. *arXiv* **2015**, arXiv:1504.06580.
20. Xu, Y.; Mou, L.; Li, G.; Chen, Y.; Peng, H.; Jin, Z. Classifying relations via long short term memory networks along shortest dependency paths. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 1785–1794.
21. Yu, X.; Lam, W. Jointly Identifying Entities and Extracting Relations in Encyclopedia Text via A Graphical Model Approach. In Proceedings of the International Conference on Coling, Beijing, China, 23–27 August 2010.
22. Zheng, S.; Hao, Y.; Lu, D.; Bao, H.; Xu, J.; Hao, H.; Xu, B. Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing* **2017**, *257*, 59–66. [CrossRef]
23. Riedel, S.; Yao, L.; McCallum, A. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 148–163.
24. Ren, X.; Wu, Z.; He, W.; Qu, M.; Voss, C.R.; Ji, H.; Abdelzaher, T.F.; Han, J. Cotype: Joint extraction of typed entities and relations with knowledge bases. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 1015–1024.