Joint Entity and Relation Extraction with a Hybrid Transformer and Reinforcement Learning Based Model

Ya Xiao,[1] **Chengxiang Tan,**[1*] **Zhijie Fan,**[2] **Qian Xu,**[1] **Wenye Zhu**[1]
[1]Tongji University, Shanghai, China
[2]The Third Research Institute of the Ministry of Public Security, Shanghai, China
{1710053, jerrytan, 1310512, 1310513}@tongji.edu.cn, aaronzfan@126.com

Abstract

Joint extraction of entities and relations is a task that extracts the entity mentions and semantic relations between entities from the unstructured texts with one single model. Existing entity and relation extraction datasets usually rely on distant supervision methods which cannot identify the corresponding relations between a relation and the sentence, thus suffers from noisy labeling problem. We propose a hybrid deep neural network model to jointly extract the entities and relations, and the model is also capable of filtering noisy data. The hybrid model contains a transformer-based encoding layer, an LSTM entity detection module and a reinforcement learning-based relation classification module. The output of the transformer encoder and the entity embedding generated from the entity detection module are combined as the input state of the reinforcement learning module to improve the relation classification and noisy data filtering. We conduct experiments on the public dataset produced by the distant supervision method to verify the effectiveness of our proposed model. Different experimental results show that our model gains better performance on entity and relation extraction than the compared methods and also has the ability to filter noisy sentences.

Introduction

Extraction of entities and relations from the massive unstructured texts has been a significant task in information extraction. The inputs are the unstructured texts, while the outputs are the triplets with pairs of entity mentions and the semantic relations between them. The extraction methods are mainly divided into two categories, the pipelined methods and joint extraction methods.

Pipelined methods treat the extraction as two separate subtasks: recognizing the entities first (Nadeau and Sekine 2007) and then extracting the relations between them (Hasegawa, Sekine, and Grishman 2004). This framework is flexible and simple but has several problems. The errors from entity recognition may delivery to the relation extraction part (Li and Ji 2014), and this separated framework also ignore the interrelationships between two subtasks. On the contrary, the joint framework extracts the entities and relations with one single model; it can take advantage of the relationships between two subtasks, and thus achieves better results. The joint framework falls into one of two categories, feature-based statistical methods (Li and Ji 2014; Miwa and Sasaki 2014; Ren et al. 2017) and neural network based methods (Miwa and Bansal 2016; Zheng et al. 2017; Wang et al. 2018). The statistical methods need to extract various features which are time consuming; some global features, such as the part-of-speech tags, are usually tagged by the third-party toolkits, which may cause error propagation problems. The neural network based methods usually realize joint learning by parameter sharing to interconnect two subtasks. Particularly, Zheng et al. (2017) proposed a novel tagging schema to convert joint extraction into a tagging problem. This end-to-end neural network model can directly output the triplets with entities and relations.

However, to our best knowledge, none of the joint extraction methods with neural networks considered the problem of noisy data. The distant supervision has become a standard method for relation extraction and dataset generation. This method assumes that the sentences containing an entity pair must describe a relation of two entities. This assumption cannot identify the corresponding interrelationship between a relation and the sentence, thus many noisy data will be generated. The noisy data will cause error propagation problem, so it is an important task to filter the wrong labelled data during the extraction of entities and relations. Many existing work use multi-instance learning to track label ambiguity and filter noisy sentences (Wu et al. 2018; Zeng et al. 2015; Lin et al. 2016). Ren et al. (2017) firstly proposed a feature-based joint method of relation and entity extraction and coped with the noisy data problem simultaneously. They mainly concentrated on the noisy associations between relation mention and its candidate entity type. Feng et al. (2018) introduced the reinforcement learning(RL) method to help with the relation classification and instance selection, and proved the efficiency of using RL to select instances. Similar to the instance selection proposed by Feng et al., our joint extraction and noisy filtration tasks also has the *trial-and-error-search* and *delayed* features, which exactly fit the RL process.

*Corresponding author.

To address the issues of joint extraction and noisy filtration, we propose a hybrid deep neural network model in this paper. The joint extraction issue is addressed by sharing the parameters and interactions between two subtasks. The noisy data issue is handled by a reinforcement learning agent. Our proposed model allows joint extraction and noisy sentence filtration in one model by using transformer encoder, LSTM decoder and reinforcement learning. Our model introduces a new attenuation coefficient based position embedding to enhance the word embedding, and extracts the entities existing in a sentence with a transformer encoder and LSTM decoder. Then the embedding of extracted entities and the hidden layer of transformer encoder are treated as the input state of reinforcement learning agent to help with relation extraction and noisy sentence filtration.

Our contributions are summarized as follows:

- We propose a new model for joint entity and relation extraction, which implement the entity extraction, relation classification and noisy data filtration in one model with the help of transformer encoder.
- We formulate the relation classification and noisy data filtration as a reinforcement learning problem, which combines the sentence context information, entity information and filtered noisy data all together.

Related Works

The pipelined methods for entity recognition and relation extraction treat these two tasks separately. Recognizing an named entity, such as name, location, organization or time, is always treated as a sequence labeling problem which needs to recognize both the boundary and the category of the entity (Nadeau and Sekine 2007); the relation extraction is usually regarded as a classification task and can be implemented by the neural network models such as Convolution Neural Network(CNN) (Liu et al. 2013) or Recurrent Neural Network(RNN) (Zhou et al. 2016; Wang et al. 2016). The joint methods use one model to extract entities and relations simultaneously. The Long Short Term Memory network(LSTM) can keep long term memory by training proper gating weights thus shows a powerful capacity on many natural language processing tasks. Ren et al. (2017) presented a distant supervised framework for joint extraction of entities and relations. Miwa and Bansal (2016) also employed the end-to-end method and used the dependency tree for relation classification. Zheng et al. (2017) put forward a novel tagging schema to tag the entity related words and turn the extraction problem into a tagging problem, they also adopted a bidirectional LSTM encoder and an LSTM decoder.

The limited amount of annotated corpus for entity and relation extraction is not enough, thus, the distant supervised methods for relation extraction are proposed (Mintz et al. 2009; Ren et al. 2017; Tang et al. 2017). The distant supervision method maps the known entities in third party knowledge bases to the entities in unstructured texts to automatically generate tagged corpus. It assumes that sentences mentioning one entity pair are all describing one of the relations of the entity pair, thus can not identify the mapping relationship between a sentence and a relation, thus may produce noisy data. To cope with the noisy labeling problem, Lin et al. (2016) proposed the sentence-level attention mechanism to select the instance. Lin et al. (2017) introduced the entity description information in knowledge base to enhance the learning of entity embedding.

With the propose of the pre-trained model, such as GPT (Radford et al. 2018) and BERT (Devlin et al. 2019), researchers have obtained great improvement by using the pre-trained model in different natural language processing tasks. It has also been proved that the transformer used in GPT and BERT has better ability on extracting features than the LSTM. Alt, Hübner, and Hennig (2019) have utilized a transformer decoder in GPT for distantly supervised relation extraction and obtained state-of-the-art results.

Recent works have also introduced reinforcement learning into information extraction or noise filtering. Feng et al. (2017) used RL to jointly identify the entity mentions and relation types, but they treated the two tasks separately and use an LSTM as the feature extractor. Takanobu et al. (2019) also used RL in the joint extraction task. They designed a hierarchical paradigm with high-level relation detection and low-level entity extraction, but they do not consider the noisy data problem. Feng et al. (2018) proposed a new model for relation extraction which consists of an instance selector with RL and a relation classifier with CNN. Qin, Xu, and Wang (2018) also proposed a deep RL framework for distant supervision relation extraction. Previous methods have concentrated on joint extraction or filter noisy data in relation classification, but not considered the noisy data filtering in the joint extraction tasks.

Proposed Model

We propose a new hybrid framework for joint extraction of entities and relations, which is also able to deal with the noisy data. Figure 1 illustrate the framework of the proposed model. The model mainly consists of three parts: the transformer encoding module, the LSTM decoding module for entity detection, and the reinforcement learning agent for relation classification and noisy sentence filtering. The transformer encoding module, which contains the word embedding layer and transformer encoder, is shared by the entity detection and relation classification module. The embedding result of entity detection also helps with the relation classification module, which extracts the relations and also filter the noisy sentences. Finally, the reward from reinforcement learning is used to optimize the parameters of other modules.

Embedding layer

The embedding layer converts the word with raw input to an embedding vector. Given a sentence with n words $S = \{x_1, x_2, ..., x_n\}$, each word x_i is converted into a real valued vector x_i'. The word vectors are encoded by an embedding matrix. The traditional embedding for the encoder input consists of token embedding, mask embedding and position embedding, as shown in figure 1. The token embedding usually stands for the embedding of the basic unit of the input sentence, e.g., word for English and character for Chinese. Mask embedding is used to identify whether the token is a

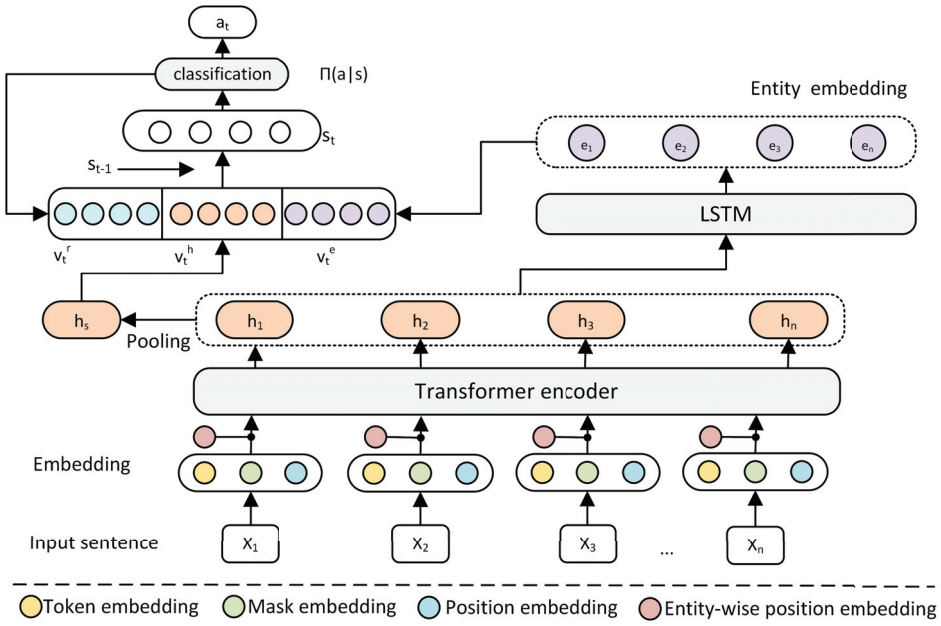

Figure 1: The framework of the proposed hybrid model.

padded one. The position embedding describes the location of each token in the input sentence.

In the joint extraction of entity and relation task, the distance between each token and the source or target entity may have an impact on the relation extraction task (Yuan et al. 2019), and the traditional position embedding can not properly demonstrate this relationship. To better illustrate the impact of distance between tokens and entities, we introduce the attenuation coefficient based position weight, named as entity-wise position embedding in figure 1. Tokens which close to the source and target entities often contain more information related to the relations. We define the position weight of each token as λ_i, calculated as:

$$\lambda_i = W_{pe} \sum_j (1 - \frac{|d_{ij}|}{D}). \quad (1)$$

In this paper, we only consider the situation that a sentence contains only one triplet, i.e., one relation type and two entities. When $j = 1$, d_{i1} is the relative distance between token x_i and the source entity, when $j = 2$, d_{i2} is the distance about the target entity. D is referred to the longest distance between token and entities. W_{pe} is a training parameter and updates as the training process; this is mainly used to cope with the situation when the test data can not see the relative distance information. We use $u_i = \lambda_i x'_i$ to denote the weighted token embedding, and the input sentence is represented as $u^S = \{u_1, u_2, ..., u_n\}$ after the embedding layer.

Transformer encoder

The transformer encoder represents words as a linear sequence with the word representations from the embedding layer. The transformer encoder contains various transformer encoding layer, while each layer contains multi-head attention, layer normalization, feed forward and layer normalization. The output of the layer normalization is regarded as the output of the encoding layer. The attention layer maps a query(Q), a set of key(K)-value(V) pairs to an output, the output is computed as a weighted sum of the values. Query, key and value are matrices which are calculated by multiplying the input embedding u by the trained weight matrices W, for example, $Q = W^Q U$. The attention weighted value is described as Equation (2), where d_k is the dimension of keys and values.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \quad (2)$$

Different from the sequential bidirectional LSTM which can only calculate the forward or backward hidden states and concat them together, the transformer encoder can merge both left and right contexts at the same time and the attention of each word can be calculated simultaneously. Besides, since the transformer encoder usually has much more layers, the feature extraction ability is much stronger than BiLSTM.

Entity detection

We treat the entity detection as a sequence labeling task with an LSTM decoder. We use the entity tags proposed by Takanobu et al. (2019) as the tag scheme because they described not only the position of words in each entity but also the source and target entities for a relation. The space of entity tags is denoted as: $\{S, T, O\} \times \{B, I\} \cup \{N\}$, where S represents the source entity, T is the target entity, O is the entities that are not associated with the predicted relation type. B and I indicate the word is the beginning or inside of an entity. N stands for the non-entity words.

The LSTM structure is used to obtain the entity tags. For token u_t, the input of the LSTM decoder layer includes: h_t from the transformer encoding layer, former hidden state h_{t-1} of the LSTM layer, and the former predicted entity tag vector e_{t-1}. The vector e_t is the predicted tag vector. The detailed operations are as follows:

$$i_t = \sigma(W_{xi}u_t + W_{hi}h_{t-1} + W_{ci}e_{t-1} + b_i), \quad (3)$$
$$f_t = \sigma(W_{xf}u_t + W_{hf}h_{t-1} + W_{cf}e_{t-1} + b_f), \quad (4)$$
$$g_t = \tanh(W_{xc}u_t + W_{hc}h_{t-1} + W_{cc}e_{t-1} + b_c), \quad (5)$$
$$c_t = i_t g_t + f_t c_{t-1}, \quad (6)$$
$$o_t = \sigma(W_{xo}u_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o), \quad (7)$$
$$h_t = o_t \tanh(c_t), \quad (8)$$
$$e_t = W_{es}h_t + b_{es}. \quad (9)$$

A softmax layer is implemented to compute the entity tag probabilities based on e_t, denoted as:

$$p_t^i = softmax(w_e e_t + b_e). \quad (10)$$

Reinforcement Learning Based Relation Classification

A reinforcement learning agent is deployed for relation classification. By using the policy and reward mechanism, the model can also filter noisy sentences generated by distant supervised learning. As shown in the upper-left corner of Figure 1, the reinforcement learning based relation classification take the pooled hidden layer of the transformer encoder, the active entity embedding of the LSTM decoding layer, the labelled noisy data and the data from the previous time step as the input of each state. Then the agent follows a policy function to take actions, including classifying the relation types and determining whether the input sentence is noisy or not. The reward will be received from the action results and to optimize the policy function. We will introduce the action, state, policy and reward in this module as follows.

Action At each time step, the agent aims to extract the relation type of the input sentence and to determine whether the sentence is noisy for the corresponding relation, i.e., whether the relation is actually described in the input sentence. The action a_t is selected from action set $A = \{R\} \times \{0, 1\}$, where $\{R\}$ is a set of relation types, 0 means the sentence is not noisy and has higher weight when calculating the loss function; 1 represents the sentence is noisy and should have lower weight for the loss function. The relation action is used to calculate the rewards and noisy action will affect the loss and the state of the next step.

State The state s_t represents the information of the input sentence at current time step t. To take advantage of the recognized entities features, origin sentence features and the features of the noisy sentences, the current state takes the entity embedding, hidden state of the encoding layer, and the vector of noisy sentences into consideration. The representation of s_t is as follows:

$$s_t = f(W_s[v_t^e, v_t^h, v_t^r, s_{t-1}]), \quad (11)$$

where $f(\cdot)$ is a function implemented by a multilayer perception. W_s is the matrix of weighted parameters. v_t^e represents the active entity embedding, since some words are non entity words and may be less useful in the relation extraction step, we only use the average vector of the source and target entity embedding as the active embedding; v_t^h is the pooled sentence embedding of the transformer encoding layer, similar to the previous model (Devlin et al. 2019), we use the hidden embedding of the first token to represent the entire sentence; v_t^r indicates the average vector of the noisy sentences in the early states.

Policy The relation and noisy data classification policy is implemented by a softmax function, described as follows:

$$a_t \sim \pi(a_t|s_t) = softmax(W_t(\delta s^t) + b_t), \quad (12)$$

where the δ helps adjust the weight of different parts of s^t.

Reward The action of our reinforcement learning agent contains two parts, i.e., classification of relation and noisy sentences. The noisy classification is used to improve the performance of relation classification, therefore, the reward of the agent is denoted as r_t, which represents the reward by the performance of the entity and relation classification. Inspired by Qin, Xu, and Wang(2018), the relation classification performance is measured by the difference of the F-score for the selected sentences, and the entity detection performance is measured according to the predict precision of target entities, as shown in Equation (13).

$$r_t = Norm(F^i - F^{i-1}) + Norm(\sum_i r^e), \quad (13)$$

where i represents different sets of sentences. For example, we divide the input sentences into various sets, and calculate the value of F-score between two neighbor sets, if the F-score is improved, the reward is positive, meaning that the noisy sentences are sort of correctly classified; if the F-score is decreased, which means the agent performs worse and the reward is negative. $Norm$ function is used to convert the reward score to a rational numeric range. The F-score(*F*) is the harmonic mean of precision(*P*) and recall(*R*), we use the F1-score in this paper, calculated by:

$$F = \frac{2 \times P \times R}{P + R}. \quad (14)$$

The reward for entity detection r^e equals l when the word is corrected tagged and is a source or target entity; r^e equals m when the word is corrected tagged and is a non-related entity; r^e equals n when the word is corrected tagged and is not an entity; r^e equals p when the word is wrong labelled; where $l > m > n > 0 > p$.

Model Training

For the entity detection module, we define the objective function using cross-entropy as follows:

$$J(\Phi) = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{N}\log(p_i^j|x_i^j, \Phi), \quad (15)$$

where M is the number of training set of training process, and N is the length of training sentence.

To optimize the policy function of reinforcement learning based relation classification, we aim to maximize the expect

total reward, which can be decomposed into two parts, immediate reward and the discounted value of successor state. The objective function is defined as:

$$\begin{aligned} J(\Theta) &= R(s_t, a_t) \\ &= \mathbb{E}_{\mathbf{s},a}[\sum_{j=0}^{T-1} \gamma^j r_{t+j} + \gamma^T R(s_{t+T}, a_{t+T})|s_t, a_t], \end{aligned} \tag{16}$$

where s_t and a_t are denoted as Equation (11) and (12) respectively. T is the number of time steps. The gradient of objective function is calculated according to the policy gradient method (Sutton et al. 2000) and the reinforce algorithm (Williams 1992):

$$\nabla_\Theta J(\Theta) = \mathbb{E}_{\mathbf{s},a}[R(s_t, a_t)\nabla_\Theta \log \pi_\Theta(s_t, a_t)]. \tag{17}$$

The entire training process of the proposed hybrid model is described in Algorithm 1.

Algorithm 1 Training process for the proposed hybrid model

INPUT: Training data $\mathbf{B} : \{B^1, B^2, ..., B^N\}$
for $epoch\ i = 1 \rightarrow E$ **do**
  for $B^k \in \mathbf{B}$ **do**
    Calculate hidden state h_t according to the transformer encoding layer
    Calculate entity embedding e_t according to the entity detection layer
    for each sentence x in B^k **do**
      Calculate state s_t by Equation (11)
      Sample action a_t by Equation (12)
      Obtain reward r_t by Equation (13)
    end for
    Jointly optimize entity detection and relation extraction module with Equation (15) and (17)
  end for
end for

Experiments

Experimental Setting

Dataset We evaluate our proposed model on a widely used dataset which is developed by the distant supervised method and the test set is manually annotated (Hoffmann et al. 2011). Similar to Takanobu et al. (2019), we also filtered the dataset by removing the sentences which have no relation, i.e. the *None* relation type, and relations in training set but not in the test set. There are $62,648$ sentences, $74,312$ triplets in the training set and 370 sentences in the test set, denoted as *TestSet1*. The size of the relation set is 12. Because *TestSet1* is manually annotated with almost no noisy data, to better evaluate our model on both joint extraction and noisy filtration tasks, we also randomly selected 5 percent of the training data as the additional test set, denoted as *TestSet2*, which contains $3,763$ sentences. We use another 5 percent of the remaining training data as the validation set to tune the hyperparameters referring to the previous studies.

Parameter Settings Pre-training a transformer based model is computationally expensive, and our main goal is to show the effectiveness of combining pre-trained model and downstream models, so we use the pre-trained language model BERT (Devlin et al. 2019) as our pre-trained encoder. We use the BERT-base version with 12 encoder layers, 12 heads attention, 768 hidden size and about $110M$ parameters. The number of LSTM units in the decoder layer is 768. The learning rate is $5e-5$ for the encoder and entity detection module, and is set to 0.003 in the reinforcement learning module, and the batch size is 16.

Baselines We choose both the pipelined methods (*FCM*) and the joint learning methods as the baselines. For joint learning methods, we also consider the feature-based model (*CoType*) and neural network based models. Besides, we also do some ablation studies on different variants of our method to prove the effectiveness of different modules.

- **FCM**(Gormley, Yu, and Dredze 2015): a pipelined and feature-rich compositional embedding model which combines the handcrafted unlexicalized linguistic features and word embeddings for relation extraction.
- **CoType**(Ren et al. 2017): a framework that jointly represents the entity mentions, relation mentions, text features and type labels into two meaningful vector spaces which denote entities and relations respectively.
- **SPTree**(Miwa and Bansal 2016): an joint extraction model that represents the word sequence and distance features of the dependency tree with a BiLSTM-RNN structure.
- **Tagging**(Zheng et al. 2017): a joint method to extract the entities and relations simultaneously with a novel tagging schema. In this model, each word is allocated with a tag including both the entity and relation information.
- **HRL**(Takanobu et al. 2019): a hierarchical reinforcement learning based model that decomposes the joint task into a high-level relation detection and a low-level entity extraction.
- **JRE_L**: a variant of our proposed model which replace the transformer encoder with the traditional bidirectional LSTM. In this model we initialize the word vectors with 300 dimensional Glove vectors (Pennington, Socher, and Manning 2014), and the batch size is set to 128.
- **JRE_T**: a variant of our proposed model with transformer encoder but no entity-wise position embedding and reinforcement learning agent.
- **JRE_T_PE**: a variant of our proposed model with transformer encoder and entity-wise position embedding but no reinforcement learning agent.
- **JRE_TRL**: our proposed joint entity and relation extraction model with transformer encoder, entity-wise position embedding and reinforcement learning agent.

Experimental Results

To evaluate the performance of our proposed model, we use Precision(*P*), Recall(*R*) and F-score(*F*) as the evaluation metrics in the joint entity and relation extraction.

Table 1: Results on entity and relation detection respectively.

Corpus	Methods	Entity			Relation		
		P	*R*	*F*	*P*	*R*	*F*
TestSet1	FCM	-	-	-	0.502	0.479	0.490
	CoType	0.659	0.620	0.639	0.558	0.558	0.558
	SPTree	0.514	0.442	0.470	0.650	0.614	0.631
	HRL	**0.737**	0.664	0.699	0.676	0.676	0.676
	JRE_L	0.691	0.593	0.638	0.553	0.567	0.560
	JRE_T	0.715	0.627	0.668	0.610	0.584	0.597
	JRE_T_PE	0.720	0.681	0.698	**0.710**	0.641	0.645
	JRE_TRL	0.728	**0.720**	**0.725**	0.692	**0.683**	**0.686**
TestSet2	FCM	-	-	-	0.445	0.362	0.399
	CoType	0.667	0.647	0.657	0.580	0.486	0.529
	SPTree	0.457	0.399	0.415	0.628	0.489	0.548
	HRL	0.696	0.654	0.674	0.626	0.588	0.606
	JRE_L	0.644	0.556	0.597	0.542	0.445	0.489
	JRE_T	0.652	0.587	0.618	0.578	0.572	0.575
	JRE_T_PE	0.698	0.628	0.661	0.593	0.606	0.598
	JRE_TRL	**0.720**	**0.695**	**0.708**	**0.615**	**0.630**	**0.623**

Entity and relation extraction results Table 1 compares our model with the baseline methods on both entity detection and relation classification with two test datasets. We calculate the metrics for each label, and find their average weighted by the number of true instances for each label. For entity detection, we only measure the prediction of entity tags and ignore the non-entity tags. Precision(*P*) measures the percentage of correctly predicted entity tags; Recall(*R*) measures whether the actual entity tags are correctly predicted. For relation classification, *P* denotes the percentage of correctly predicted relation class, *R* denotes whether the actual relation types are correctly predicted.

FCM is a pipelined method which suppose the entities are already tagged, so we only compare the performance of relation extraction for *FCM*. It can be seen that our method (*JRE_TRL*) outperforms other methods on both entity detection and relation classification tasks at *R* and *F* metrics. Although *HRL* has a higher precision on entity detection, we have got better recall and F-score. The surpass may because that *HRL* optimizes the reinforcement learning agent based on the performance of entity and relation classification on each word, which is very time consuming. All the joint methods perform better than the pipelined method (*FCM*) proves the importance of considering the interactions between two tasks. The neural models (*SPTree*, *HRL*) are more efficient than the feature-based method (*CoType*) on relation extraction, this is because that the neural models can take advantage of various kinds of features, including word embedding and location features. Both *SPTree* and *JRE_L* uses an BiLSTM-RNN structure. But *SPTree* get worse results compared to *JRE_L* in entity detection task for the lack of the LSTM decoder. But *SPTree* performs better on relation detection, because it utilizes not only the vector feature of words, but also the linguistic features, such as the part-of-speech tag, chunks and syntactic parsing tree.

JRE_T performs better than *JRE_L* proves that transformer encoder has better ability on extracting the hidden feature than the BiLSTM. *JRE_T_PE* with entity-wise position embedding receives much higher relation classification results than *JRE_T* demonstrates that the distance between each word and entities do affect the relation type. The better performance on entity detection also indicates that the parameter in *JRE_T_PE* can properly learn the entity related position information from the training data. The proposed model *JRE_TRL* improves little or even slightly lower than *JRE_T_PE* on *TestSet1*, this is because the *TestSet1* is manually annotated and contains almost no noisy sentences, leading the reinforcement learning agent not fully works. *JRE_TRL* improves a lot on *TestSet2* than *JRE_T_PE* shows that by filtering the noisy sentences, the model can learn better information related to the relation, thus can improve the performance.

Table 2: Results on the joint entity and relation detection.

Methods	P	R	F
FCM	0.432	0.294	0.350
CoType	0.486	0.386	0.430
SPTree	0.522	0.541	0.531
Tagging	0.469	0.489	0.479
HRL	0.538	0.538	0.538
JRE_TRL	**0.542**	**0.542**	**0.542**

Joint extraction results To thoroughly evaluate the proposed model, we further analyze the joint extraction performance, as shown in Table 2. A prediction is treated as correct if both the entity and relation are correctly predicted. The results are similar to those in Table 1, the joint methods perform better than the pipelined method, and neural models are more efficient than the feature-based method. *SPTree* also performs good on the recall may because they use many other resources, especially the dependency tree in the relation extraction. But this method may cause error

Table 3: Some examples of noisy samples detected by our noisy data filtering module.

Relation	**/people/person/place_lived**
	And they would never understand why, for **Bill Elliott**, there was no joy in **Dawsonville**.
Relation	**/location/location/contains**
	The museum, which opened in the **North Beach** neighborhood of **San Francisco** last fall, has made plans to sponsor a spring tour of Mr. Morgan's space scrap in a vintage Airstream trailer, creating a kind of Electric Kool-Aid Acid Test for the astronomy set.
	Winter Park, the fourth-largest ski resort in **Colorado** after **Vail**, Keystone and Snowmass , has a Ski for Free program , in which volunteers help to operate the resort 's racing events in return for free skiing .

propagation and not suitable for the language with inadequate dependency tree resources. The only minor surpass of *JRE_TRL* compared to *HRL* is because that *HRL* takes fully consideration of the word-level information and performs reinforcement learning on each word. However, we believe that the pre-trained language models, such as BERT used in our model, have learned adequate word-level features from the massive training data; it is less meaningful to spend too much time and computing resources on the word-level features in the downstream tasks.

The outperforms of our proposed model on different aspects of entity and relation extraction proves the efficiency of the hybrid transformer and reinforcement learning based method.

Noisy data filtering To measure the performance of data filtering in the reinforcement learning agent, we manually annotated 505 sentences, to decide whether the corresponding relation is actually described in the sentences. We use the filtering results from the reinforcement learning agent to compare with the annotated sentences. The 505 annotated sentences contain 310 truly noisy and 195 not noisy. We define noisy as positive and not noisy as negative here. The prediction results after the reinforcement learning agent are as follows: true positive: 180, false negative: 130, false positive: 75, true negative: 120. The precision of noisy data filtering is $180/255 = 70.58\%$, and recall is $180/310 = 58.06\%$, which demonstrate that our model can properly identify the noisy sentences. In Table 3, we present some examples of noisy sentences which are detected by our noisy data filtering agent. These sentences are annotated as a specific relation type in the dataset, as listed in the relation row. However, they do not actually describe that relation. For example, there is not any valuable information reflect that the entity *Bill Elliott* has lived in *Dawsonville*.

Performance of the reinforcement learning agent The aim of the reinforcement learning agent is to optimize the policy function and increase the total reward. To evaluate the performance, we calculate the total reward by Equation (13) after each batch of the training process, for better visualization, we split the reward values into $1,200$ parts and calculate the mean of each part, as shown in Figure 2. It can be seen that the reward begins from a small value, then increases as the training iterations, and finally reach a relatively stable value. The result proves that the proposed reinforcement learning method is useful on improving the rewards of the joint extraction task.

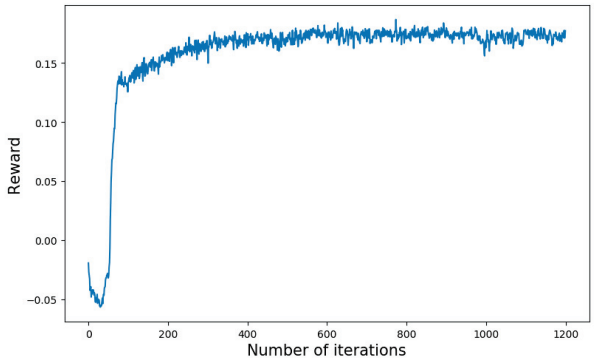

Figure 2: Change of total rewards of the proposed model.

Conclusion

In this study, we aim at the problem of jointly extracting the entities and relations on noisy data and propose a hybrid transformer and reinforcement learning based model. We use the attenuation coefficient based position embedding to weight the word vectors for better utilizing the distance between words and entity mentions. A transformer encoder is deployed to take full advantage of the contextual information and an LSTM decoder layer is used to classify the entity types. Finally, a reinforcement learning agent is implemented for relation classification and noisy sentence selection. The pooled hidden state from the encoder, active entity embedding from decoder, the removed noisy sentence embedding and the state of previous time step are all taken as the state input of the reinforcement learning agent. The model is optimized by maximizing the total rewards of entity detection and relation classification. Experiments are launched at the following various aspects with a commonly used dataset: entity detection, relation classification, joint extraction of entities and relations, noisy sentence filtering, and change of reward values in the reinforcement learning process. Results show that our proposed hybrid model performs better on relation and entity extraction compared to the baseline methods. Our model also has the ability to filer noisy sentences. As future works, we will enhance our model for coping with the overlapped and multiple relations.

Acknowledgments

This work is supported in part by the National Key R & D Program of China under Grant 2017YFC0803700, 2016YFB0801100 and 2017YFB0802300.

References

Alt, C.; Hübner, M.; and Hennig, L. 2019. Fine-tuning pre-trained transformer language models to distantly supervised relation extraction. In *ACL*, 1388–1398.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 4171–4186.

Feng, Y.; Zhang, H.; Hao, W.; and Chen, G. 2017. Joint extraction of entities and relations using reinforcement learning and deep learning. *Computational intelligence and neuroscience* 2017.

Feng, J.; Huang, M.; Zhao, L.; Yang, Y.; and Zhu, X. 2018. Reinforcement learning for relation classification from noisy data. In *AAAI*, 5779–5786.

Gormley, M. R.; Yu, M.; and Dredze, M. 2015. Improved relation extraction with feature-rich compositional embedding models. In *EMNLP*, 1774–1784.

Hasegawa, T.; Sekine, S.; and Grishman, R. 2004. Discovering relations among named entities from large corpora. In *ACL*.

Hoffmann, R.; Zhang, C.; Ling, X.; Zettlemoyer, L.; and Weld, D. S. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *NAACL-HLT*, 541–550.

Ji, G.; Liu, K.; He, S.; and Zhao, J. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, 3060–3066.

Li, Q., and Ji, H. 2014. Incremental joint extraction of entity mentions and relations. In *ACL*, volume 1, 402–412.

Lin, Y.; Shen, S.; Liu, Z.; Luan, H.; and Sun, M. 2016. Neural relation extraction with selective attention over instances. In *ACL*, volume 1, 2124–2133.

Liu, C.; Sun, W.; Chao, W.; and Che, W. 2013. Convolution neural network for relation extraction. In *International Conference on Advanced Data Mining and Applications*, 231–242. Springer.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, 1003–1011.

Miwa, M., and Bansal, M. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*, volume 1, 1105–1116.

Miwa, M., and Sasaki, Y. 2014. Modeling joint entity and relation extraction with table representation. In *EMNLP*, 1858–1869.

Nadeau, D., and Sekine, S. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1):3–26.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543.

Qin, P.; Xu, W.; and Wang, W. Y. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *ACL*, 2137–2147.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training.

Ren, X.; Wu, Z.; He, W.; Qu, M.; Voss, C. R.; Ji, H.; Abdelzaher, T. F.; and Han, J. 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *WWW*, 1015–1024.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Takanobu, R.; Zhang, T.; Liu, J.; and Huang, M. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *AAAI*, volume 33, 7072–7079.

Tang, S.; Zhang, J.; Zhang, N.; Wu, F.; Xiao, J.; and Zhuang, Y. 2017. Encore: External neural constraints regularized distant supervision for relation extraction. In *SIGIR*, 1113–1116. ACM.

Wang, L.; Cao, Z.; De Melo, G.; and Liu, Z. 2016. Relation classification via multi-level attention cnns. In *ACL*, volume 1, 1298–1307.

Wang, S.; Zhang, Y.; Che, W.; and Liu, T. 2018. Joint extraction of entities and relations based on a novel graph scheme. In *IJCAI*, 4461–4467.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Wu, J.; Pan, S.; Zhu, X.; Zhang, C.; and Wu, X. 2018. Multi-instance learning with discriminative bag mapping. *IEEE Transactions on Knowledge and Data Engineering* 30(6):1065–1080.

Yuan, C.; Huang, H.; Feng, C.; Liu, X.; and Wei, X. 2019. Distant supervision for relation extraction with linear attenuation simulation and non-iid relevance embedding. In *AAAI*, volume 33, 7418–7425.

Zeng, D.; Liu, K.; Chen, Y.; and Zhao, J. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, 1753–1762.

Zheng, S.; Wang, F.; Bao, H.; Hao, Y.; Zhou, P.; and Xu, B. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *ACL*, volume 1, 1227–1236.

Zhou, P.; Shi, W.; Tian, J.; Qi, Z.; Li, B.; Hao, H.; and Xu, B. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *ACL*, volume 2, 207–212.