A Hierarchical Framework for Relation Extraction with Reinforcement Learning

Ryuichi Takanobu[1,3*], **Tianyang Zhang**[1,3*], **Jiexi Liu**[2,3*], **Minlie Huang**[1,3†]
[1] Dept. of Computer Science & Technology, [2] Dept. of Physics, Tsinghua University, Beijing, China
[3] Institute for Artificial Intelligence, Tsinghua University (THUAI), China
[3] Beijing National Research Center for Information Science & Technology, China
{gxly15, zhang-ty15, liujx15}@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn

Abstract

Most existing methods determine relation types only after all the entities have been recognized, thus the interaction between relation types and entity mentions is not fully modeled. This paper presents a novel paradigm to deal with relation extraction by regarding the related entities as the arguments of a relation. We apply a hierarchical reinforcement learning (HRL) framework in this paradigm to enhance the interaction between entity mentions and relation types. The whole extraction process is decomposed into a hierarchy of two-level RL policies for relation detection and entity extraction respectively, so that it is more feasible and natural to deal with overlapping relations. Our model was evaluated on public datasets collected via distant supervision, and results show that it gains better performance than existing methods and is more powerful for extracting overlapping relations[1].

Introduction

Extracting entities, relations, or events from unstructured texts is crucial for building large-scale, reusable knowledge which can facilitate many other tasks (**?**; **?**), including knowledge base construction (**?**; **?**), question answering (**?**), and biomedical text mining (**?**).

The task of relation extraction is to identify relations (e_s, r, e_t)[2], a triple consisting of a relation type r, a source entity e_s and a target entity e_t. In this paper, we propose a novel joint extraction paradigm in the framework of hierarchical reinforcement learning (**?**), where we first detect a relation and then extract the corresponding entities as the argument of a relation.

Our model detects **relation indicators** by a high-level reinforcement learning (RL) process and identifies the participating entities for the relation by a low-level RL process. As shown in Figure 1, the extraction process makes sequential scans from the beginning to the end of a sentence (**I**). The high-level process is to detect a relation indicator at some particular position. If a certain relation is identified, a low-level sequential process is triggered to identify the corresponding entities for that relation (**II**). When the low-level subtask for entity extraction is completed (**III**), the high-level RL process continues its scan to search for the next relation (**IV**) in the sentence.

This paradigm has strengths in dealing with two issues existing in prior studies. **First**, most traditional models (**?**; **?**; **?**) determine a relation type only after all the entities have been recognized, whereas the interaction between the two tasks is not fully captured. In some sense, these methods are aligning a relation to entity pairs, and therefore, they may introduce additional *noise* since a sentence containing an entity pair may not truly mention the relation (**?**), or may describe multiple relations (**?**).

Second, there still lacks the elegance of the joint extraction method to deal with *one-to-many problems* (**overlapping relations**): one entity may participate in multiple relations in the same sentence (see *Steve Blichick* in Figure 1), or even the same entity pair within a sentence is associated with different relations. To our best knowledge, CopyR (**?**) is the only method that discussed this issue, which views relation extraction as a triple generation process. However, this method, as our experiments reveal, strongly relies on the training data, and cannot extract multi-word entity mentions.

In our paradigm, the **first issue** is handled by treating entities as the arguments of a relation. The dependency between entity mentions and relation types is formulated through designing the state representations and rewards in the high-level and low-level RL processes. The interaction is well captured since the main task (high-level RL process for relation detection) passes messages when launching a subtask (low-level RL process for entity extraction), and the low-level rewards, signifying how well a subtask is completed, are passed back to the main task. In this manner, the interaction between relation types and entity mentions can be better modeled.

The **second issue** is addressed by our hierarchical structure. By decomposing relation extraction into a high-level task for relation detection and a low-level task for entity extraction, multiple relations in a sentence can be handled separately and sequentially. As shown in Figure 1, the first relation is extracted when the main task detects the first

[*] All authors contributed equally to this work.

[†] Corresponding author: Minlie Huang.

[1] Data and code are publicly available at: https://github.com/truthless11/HRL-RE.

[2] Throughout this paper, a relation refers to a triple (e_s, r, e_t), a relation type refers to r.

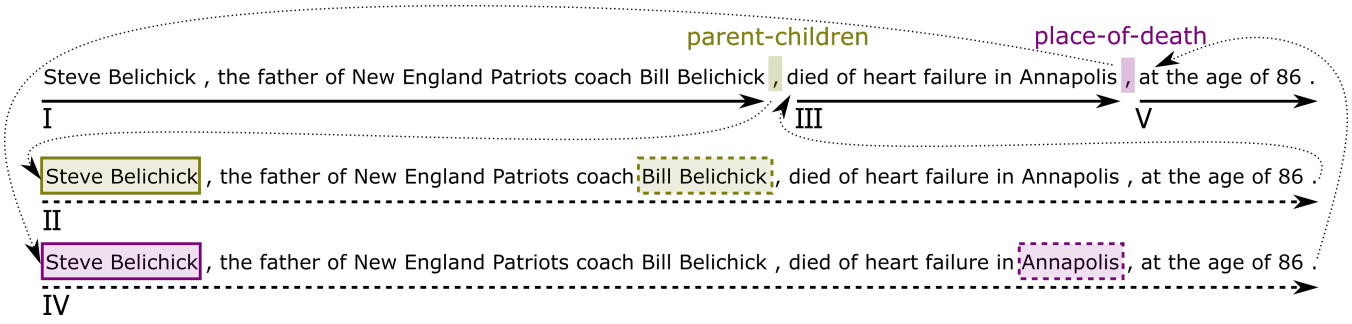

Figure 1: An example sentence which has two **overlapping relations** (*Steve Belichick*, `parent-children`, *Bill Belichick*), (*Steve Belichick*, `place-of-death`, *Annapolis*). The solid arrow indicates the high-level relation detection process, and the dashed arrow for low-level entity extraction. The dotted arrow marks a transition between the two processes. This example shows how overlapping relations are extracted (*Steve Blichick* is included in both triples).

relation type (`parent-children`), and the second relation is subsequently extracted when the second relation type (`place-of-death`) is triggered, even though the two relations share the same entity (*Steve Blichick*). Experiments demonstrate the proposed paradigm achieves strong performance over the baselines in extracting overlapping relations.

In summary, our contributions are in two folds:

- We design a novel end-to-end hierarchical paradigm to jointly identify entity mentions and relation types, which decomposes the task into a high-level task for relation detection and a low-level task for entity extraction.
- By incorporating reinforcement learning into this paradigm, the proposed method outperforms baselines in modeling the interactions between the two tasks, and extracting overlapping relations.

Related Work

Traditional pipelined approaches treat entity extraction and relation classification as two separate tasks (**?**; **?**; **?**). They first extract the token spans in the text to detect entity mentions, and then discover the relational structures between entity mentions. Although it is flexible to build pipelined methods, these methods suffer from *error propagation* since downstream modules are largely affected by the errors introduced by upstream modules.

To address this problem, a variety of joint learning methods was proposed. **?** (**?**) proposed a card-pyramid graph structure for joint extraction, and **?** (**?**) developed graph-based multi-instance learning algorithms. However, the two methods both applied a greedy search strategy to reduce the exploration space aggressively, which limits the performance. Other studies employed a structured learning approach (**?**; **?**). All these models depend on *heavy feature engineering*, which requires much manual efforts and domain expertise.

On the other hand, **?** (**?**) proposed to first extract **relation triggers**, which refer to a phrase that **explicitly** expresses the occurrence of a relation in a sentence, and then determine their arguments to reduce the task complexity. Open IE systems *ReVerb* (**?**) identifies relational phrases using lexical constraints, which also follows a "relation"-first, "argument"-second approach. But there are many cases where no relation trigger appears in a sentence so that such relations cannot be captured in these methods.

Neural models for joint relation extraction are investigated in recent studies (**?**; **?**). **?** (**?**) proposed a neural model that shares parameters for entity extraction and relation classification, but the two tasks are separately handled, and the final decision is obtained via exhaustively enumerating the combinations between detected entity mentions and relation types. Unlike aforementioned methods that all the entities are recognized first, **?** (**?**) used a tagging scheme which applies a Cartesian product of the relation type tags and the entity mention tags, and thus each word is assigned a unique tag that encodes entity mentions and relation types simultaneously. However, it is unable to deal with overlapping relations in a sentence: if an entity is the argument of multiple relations, the tag for the entity should not be unique. The recent study (**?**) is closely related to ours that aims to handle overlapping relations. It employs multiple decoders based on sequence-to-sequence (Seq2Seq) learning where a decoder copies an entity word from the source sentence and each triple in a sentence is generated by different decoders, but such a method strongly relies on the annotation of training data and it cannot extract an entity that has multiple words.

Reinforcement learning has been witnessed in information extraction very recently. RL was employed to acquire and incorporate external evidence in event extraction (**?**). **?** (**?**) used RL to train an instance selector to denoise training data obtained via distant supervision for relation classification. Improvement was reported in distant supervision relation type extraction by exploring RL to redistribute false positives into the negative examples (**?**).

Hierarchical Extraction Framework

Overview

First of all, we define *relation indicator* as follows:

Definition 1. *Relation indicator* is the position in a sentence when sufficient information has been mentioned to identify a semantic relation. Different from *relation trigger* (i.e.,

explicit relation mention), relation indicators can be verbs (e.g. *die* of), nouns (e.g. his *father*), or even prepositions (e.g. *from/by*), other symbols such as comma and period (As shown in Figure 1, the relation type `place-of-death` can be signified till the *comma* position).

Relation indicator is crucial for our model to complete the extraction task, because the entire extraction task is decomposed into relation indicator detection and entity mention extraction.

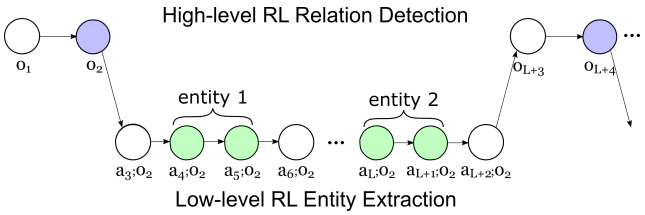

Figure 2: Overview of a hierarchical agent in relation extraction.

The entire extraction process works as follows. An agent predicts a relation type at a particular position when it scans a sentence sequentially. Note that this process of relation detection needs no annotation of entities, thus different from relation classification which is to identify the relations between pairs of entities. When there is no sufficient evidence to indicate a semantic relation at a time step, the agent may choose `NR`, a special relation type that indicates no relation. Otherwise a relation indicator is triggered, the agent launches a subtask for entity extraction to identify the arguments of the relation, the two entities. When the entity mentions are identified, the subtask is completed and the agent continues to scan the rest of the sentence for other relations.

Such a process can be naturally formulated as a semi-Markov decision process (**?**): 1) a high-level RL process that detects a relation indicator in a sentence; 2) a low-level RL process that identifies the associated entities for the corresponding relation. By decomposing the task into a hierarchy of two RL processes, the model is advantageous at dealing with sentences which have multiple relation types for the same entity pair, or one-to-many entities in which an entity is the argument of multiple relations.

Relation Detection with High-level RL

The high-level RL policy μ aims to detect the relations in a sentence $S = w_1 w_2 \cdots w_L$, which can be regarded as a conventional RL policy over options. An option refers to a high-level action, and a low-level RL process will be launched once an option is executed by the agent.

Option: The option o_t is selected from $\mathcal{O} = \{\texttt{NR}\} \cup \mathcal{R}$ where `NR` indicates no relation, and $\mathcal{R}$ is the relation type set. When a low-level RL process enters a terminal state, the control of the agent will be taken over to the high-level RL process to execute the next options.

State: The state $\mathbf{s}_t^h \in \mathcal{S}$ of the high level RL process at time step t, is represented by: 1) the current hidden state $\mathbf{h}_t$, 2) the relation type vector $\mathbf{v}_t^r$ (the embedding of the *latest* option $o_{t'}$ that $o_{t'} \neq \texttt{NR}$, a learnable parameter), and 3) the state from the last time step $\mathbf{s}_{t-1}$[3], formally represented by

$$\mathbf{s}_t^h = f^h(\mathbf{W}_s^h[\mathbf{h}_t; \mathbf{v}_t^r; \mathbf{s}_{t-1}]), \tag{1}$$

where $f^h(\cdot)$ is a non-linear function implemented by MLP. To obtain the hidden state $\mathbf{h}_t$, we introduce a sequence Bi-LSTM over the current input word embedding $\mathbf{w}_t$:

$$\begin{aligned} \overrightarrow{\mathbf{h}_t} &= \overrightarrow{LSTM}(\overrightarrow{\mathbf{h}_{t-1}}, \mathbf{w}_t), \\ \overleftarrow{\mathbf{h}_t} &= \overleftarrow{LSTM}(\overleftarrow{\mathbf{h}_{t+1}}, \mathbf{w}_t), \\ \mathbf{h}_t &= [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]. \end{aligned} \tag{2}$$

Policy: The stochastic policy for relation detection $\mu : \mathcal{S} \rightarrow \mathcal{O}$ which specifies a probability distribution over options:

$$o_t \sim \mu(o_t|\mathbf{s}_t^h) = softmax(\mathbf{W}_\mu \mathbf{s}_t^h). \tag{3}$$

Reward: Then, the environment provides intermediate reward r_t^h to estimate the future return when executing o_t. The reward is computed as below:

$$r_t^h = \begin{cases} -1, & if\ o_t\ not\ in\ S \\ 0, & if\ o_t = \texttt{NR} \\ 1, & if\ o_t\ in\ S. \end{cases} \tag{4}$$

If $o_t = \texttt{NR}$ at certain time step, the agent transfers to a new high-level inter-option state at the next time step. Otherwise the low-level policy will execute the entity extraction process. The inter-option state will not transfer until the subtask over current option o_t is done, which may take multiple time steps. Such a semi-Markov process continues until the last option about the last word w_L of S is sampled. Finally, a final reward r_{fin}^h is obtained to measure the sentence-level extraction performance that μ detects:

$$r_{fin}^h = F_\beta(S) = \frac{(1+\beta^2)Prec \cdot Rec}{\beta^2 Prec + Rec}, \tag{5}$$

where F_β is the weighted harmonic mean of *precision* and *recall* in terms of the relations in S. $Prec/Rec$ indicates precision/recall respectively, computed over one sentence.

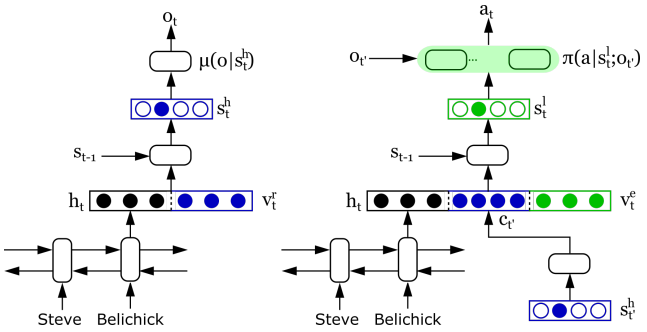

Figure 3: Illustration of a two-level hierarchical policy structure. Left panel shows the high-level policy for relation detection, and right panel shows the low-level policy for entity extraction.

[3] where $\mathbf{s}_{t-1} = \mathbf{s}_{t-1}^h$ if the agent sampled a high-level option at last time step $t-1$, and $\mathbf{s}_{t-1} = \mathbf{s}_{t-1}^l$ if the agent sampled a low-level action.

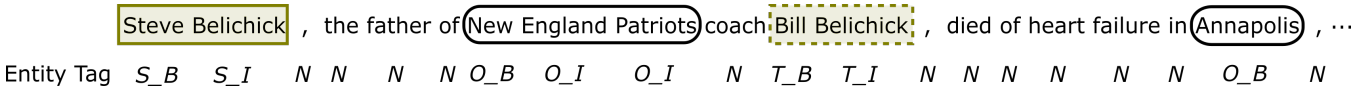

Figure 4: The entity annotation scheme for the example sentence in Figure 1 when the agent predicts a relation type `parent-children` between *Steve Belichick* and *Bill Belichick*. In this example, *New England Patriots* and *Annapolis* are not-concerned entities with respect to relation type `parent-children`.

Entity Extraction with Low-level RL

Once the high-level policy has predicted a non-NR relation type, the low-level policy π will extract the participating entities for the corresponding relation. The low-level policy over actions (primitive actions) is formulated very similarly as the high-level policy over options. To make the predicted relation type accessible in the low-level process, the option $o_{t'}$ from the high level RL is taken as additional input throughout the low-level extraction process.

Action: The action at each time step is to assign an entity tag to the current word. The action space, i.e., entity tag space $\mathcal{A} = (\{\texttt{S}, \texttt{T}, \texttt{O}\} \times \{\texttt{B}, \texttt{I}\}) \cup \{\texttt{N}\}$, where S represents the participating source entity, T for the target one, O for the entities that are not associated with the predicted relation type $o_{t'}$, and N for for non-entity words. Note that, the same entity mention may be assigned with different S/T/O tags depending on different relation types concerned at the moment. In this way, the model can deal with overlapping relations. In addition, we use the B/I symbols to represent the beginning word and the inside of an entity, respectively. Refer to Figure 4 for an example.

State: Similar to the policy for relation detection, the low-level intra-option state $\mathbf{s}_t^l$ is represented by 1) the hidden state $\mathbf{h}_t$ of current word embedding $\mathbf{w}_t$, 2) the entity tag vector $\mathbf{v}_t^e$ which is a learnable embedding of a_{t-1}, 3) the state from previous time step $\mathbf{s}_{t-1}$, and 4) the context vector $\mathbf{c}_{t'}$ using the relational state representation assigned to the latest option $\mathbf{s}_{t'}^h$ in Eq. (1), as follows:

$$\begin{aligned} \mathbf{c}_{t'} &= g(\mathbf{W}_h^l \mathbf{s}_{t'}^h) \\ \mathbf{s}_t^l &= f^l(\mathbf{W}_s^l[\mathbf{h}_t; \mathbf{v}_t^e; \mathbf{s}_{t-1}; \mathbf{c}_{t'}]), \end{aligned} \tag{6}$$

where $\mathbf{h}_t$ is the hidden state obtained from the Bi-LSTM module in Eq. (2), and $f^l(\cdot)$, $g(\cdot)$ are non-linear functions implemented by MLP. Note that $\mathbf{s}_{t-1}$ may be a state either from the high-level RL process or the low-level one.

Policy: The stochastic policy for entity extraction $\pi : \mathcal{S} \rightarrow \mathcal{A}$ outputs an action distribution given intra-option state $\mathbf{s}_t^l$ and the high-level option $o_{t'}$ that launches the current subtask.

$$a_t \sim \pi(a_t|\mathbf{s}_t^l; o_{t'}) = softmax(\mathbf{W}_\pi[o_{t'}]\mathbf{s}_t^l), \tag{7}$$

where $\mathbf{W}_\pi$ is an array of $|\mathcal{R}|$ matrices.

Reward[4]: Given the relation type $o_{t'}$, the entity tag for each word can be easily obtained by sampling actions from the policy. Therefore, an immediate reward r_t^l is provided when the action a_t is sampled by simply measuring the prediction error over gold-standard annotation:

$$r_t^l = \lambda(y_t) \cdot sgn(a_t^l = y_t(o_{t'})), \tag{8}$$

where $sgn(\cdot)$ is the sign function, and $y(o_{t'})$ is the gold-standard entity tag conditioned on the predicted relation type $o_{t'}$. Here $\lambda(y)$ is a bias weight for down-weighing non-entity tag, defined as follows:

$$\lambda(y) = \begin{cases} 1, & if\ y \neq \texttt{N} \\ \alpha, & if\ y = \texttt{N}. \end{cases} \tag{9}$$

The smaller α leads to less reward on words that are not entities. In this manner, the model avoids to learn a trivial policy that predicts all words as N (non-entity words). When all the actions are sampled, an additional final reward r_{fin}^l is computed. If all the entity tags are predicted correctly, then the agent receives +1 reward, otherwise -1.

Hierarchical Policy Learning

To optimize the high-level policy, we aim to maximize the expected cumulative rewards from the main task at each time step t as the agent samples trajectories following the high-level policy μ, which can be computed as follows:

$$J(\theta_{\mu,t}) = \mathbb{E}_{\mathbf{s}^h, o, r^h \sim \mu(o|\mathbf{s}^h)}[\sum_{k=t}^{T} \gamma^{k-t} r_k^h], \tag{10}$$

where μ is parameterized by θ_μ, γ is a discount factor in RL, and the whole sampling process μ takes T time steps before it terminates.

Similarly, we learn the low-level policy by maximizing the expected cumulative intra-option rewards from the subtask over option $o_{t'}$ when the agent samples along low-level policy $\pi(\cdot|o_{t'})$ at time step t:

$$J(\theta_{\pi,t}; o_{t'}) = \mathbb{E}_{\mathbf{s}^l, a, r^l \sim \pi(a|\mathbf{s}^l; o_{t'})}[\sum_{k=t}^{T'} \gamma^{k-t} r_k^l], \tag{11}$$

if the subtask ends at time step T'.

By decomposing the cumulative rewards into a Bellman equation, we acquire:

$$\begin{aligned} R^\mu(\mathbf{s}_t^h, o_t) =& \mathbb{E}[\sum_{j=0}^{N-1} \gamma^j r_{t+j}^h + \\ & \gamma^N R^\mu(\mathbf{s}_{t+N}^h, o_{t+N}) | \mathbf{s}_t^h, o_t], \\ R^\pi(\mathbf{s}_t^l, a_t; o_{t'}) =& \mathbb{E}[r_t^l + \gamma R^\pi(\mathbf{s}_{t+1}^l, a_{t+1}; o_{t'}) | \mathbf{s}_t^l, a_t], \end{aligned} \tag{12}$$

where N is the number of time steps that a subtask continues when the entity extraction policy runs upon option o_t , so the

[4]We only discuss the situation where $o_{t'}$ is included in S, i.e. the subtask option $o_{t'}$ is correctly predicted. Otherwise, all the low-level rewards are set to 0, which can be seen that the agent has done nothing with the low-level policy.

agent's next option is o_{t+N}. In particular, if $o_t = \texttt{NR}$, then $N = 1$.

Then, we use policy gradient methods (**?**) with the REINFORCE algorithm (**?**) to optimize both high-level and low-level policies. With the likelihood ratio trick, the gradient for the high-level policy yields:

$$\begin{aligned}\nabla_{\theta_\mu} J(\theta_{\mu,t}) =& \mathbb{E}_{\mathbf{s}^h, o, r^h \sim \mu(o|\mathbf{s}^h)}[R^\mu(\mathbf{s}_t^h, o_t) \\ & \nabla_{\theta_\mu} \log \mu(o|\mathbf{s}_t^h)],\end{aligned} \tag{13}$$

and the gradient for the low-level policy yields:

$$\begin{aligned}\nabla_{\theta_\pi} J(\theta_{\pi,t}; o_{t'}) =& \mathbb{E}_{\mathbf{s}^l, a, r^l \sim \pi(a|\mathbf{s}^l; o_{t'})}[R^\pi(\mathbf{s}_t^l, a_t; o_{t'}) \\ & \nabla_{\theta_\pi} \log \pi(a|\mathbf{s}_t^l; o_{t'})].\end{aligned} \tag{14}$$

The entire training process is described at Algorithm 1.

Algorithm 1: Training Procedure of HRL

```
Calculate h_t for each word in the sentence with Bi-LSTM ;
Initiate state s_0^h ← 0 and time step t ← 0;
for i ← 1 to Text Length do
    t ← t + 1 ;
    Calculate s_t^h by Eq. (1);
    Sample o_t from s_t^h by Eq. (3);
    Obtain reward r_t^h by Eq. (4);
    if o_t ≠ NR then
        for j ← 1 to Text Length do
            t ← t + 1 ;
            Calculate s_t^l by Eq. (6);
            Sample a_t^l from s_t^l by Eq. (7);
            Obtain reward r_t^l by Eq. (8);
        end
        Obtain low-level final reward r_fin^l;
    end
end
Obtain high-level final reward r_fin^h by Eq. (5);
Optimize the model with Eq. 13 and Eq. (14);
```

Experiments

Experimental Setting

Datasets We evaluated our model on the New York Times corpus which is developed by distant supervision and contains *noisy* relations. The corpus has two versions: 1) The original version generated by aligning the raw data with Freebase relations (**?**); 2) A smaller version of which the test set was manually annotated (**?**). We name the original version as *NYT10*, and the smaller version as *NYT11*. We split some of the training data from *NYT11* to construct *NYT11-plus*, which will be described later.

We filtered the datasets by removing 1) the relations in the training set whose relation type does not exist in the test set; 2) the sentences that contain no relations at all. Such a preprocess is also in line with the settings in the literature (for instance, *Tagging*). All the baselines are evaluated in this setting for fair comparison. The statistics of the two filtered datasets are presented in Table 1.

For each dataset, we randomly chose 0.5% data from the training set for validation.

Dataset	NYT10	NYT11
# Relation types	29	12
# Training sentences	70,339	62,648
# Training relations	87,739	74,312
# Test sentences	4,006	369
# Test relations	5,859	370

Table 1: Statistics of the datasets.

Parameter Settings All hyper-parameters are tuned on the validation set. The dimension of all vectors in Eq. (1), (2) and (6) is 300. The word vectors are initialized using Glove vectors (**?**) and are updated during training. Both *relation type vectors* and *entity tag vectors* are initialized randomly. The learning rate is $4e-5$, the mini-batch size is 16, $\alpha = 0.1$ in Eq. (9), $\beta = 0.9$ in Eq. (5), and the discount factor $\gamma = 0.95$.

Evaluation Metrics We adopted standard micro-F_1 to evaluate the performance. We compared whether the extracted entity mentions can be exactly matched with those in a relation. A triplet is regarded as correct if the relation type and the two corresponding entities are all correct.

Baselines We chose two types of baselines: one is pipelined methods (FCM), and the other is joint learning methods which include feature-based methods (MultiR and CoType) and neural methods (SPTree, Tagging and CopyR). We used open source codes and conducted the experiments by ourselves.

FCM (**?**): a compositional model that combines lexicalized linguistic contexts and word embeddings to learn representations for the substructures of a sentence in relation extraction[5].

MultiR (**?**): a typical distant supervision method performing sentence-level and corpus-level extraction, which uses multi-instance weighting to deal with noisy labels in training data.

CoType (**?**): 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 (**?**): an end-to-end relation extraction model that represents both word sequence and dependency tree structures using bidirectional sequential and tree-structured LSTM-RNNs.

Tagging (**?**): 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.

CopyR (**?**): a Seq2Seq learning framework with a copy mechanism for joint extraction, where multiple decoders are applied to generate triples to handle overlapping relations.

Model	NYT10			NYT11		
	Prec	Rec	F_1	Prec	Rec	F_1
FCM	–	–	–	.432	.294	.350
MultiR	–	–	–	.328	.306	.317
CoType	–	–	–	.486	.386	.430
SPTree	.492	.557	.522	.522	**.541**	.531
Tagging	.593	.381	.464	.469	.489	.479
CopyR	.569	.452	.504	.347	.534	.421
HRL	**.714**	**.586**	**.644**	**.538**	.538	**.538**

Table 2: Main results on relation extraction.

Model	NYT10-sub			NYT11-plus		
	Prec	Rec	F_1	Prec	Rec	F_1
FCM	–	–	–	.234	.199	.219
MultiR	–	–	–	.241	.214	.227
CoType	–	–	–	.291	.254	.271
SPTree	.272	.315	.292	**.466**	.229	.307
Tagging	.256	.237	.246	.292	.220	.250
CopyR	.392	.263	.315	.329	.224	.264
HRL	**.815**	**.475**	**.600**	.441	**.321**	**.372**

Table 3: Performance comparison on extracting overlapping relations.

Main Results

The results on relation extraction are presented in Table 2. Noticeably, there is a significant gap between the performance on noisy data (*NYT10*) and that on clean data (*NYT11*) as all the models are trained on noisy data. It can be seen that our method (HRL) outperforms the baselines on the two datasets. Significant improvements can be observed on *NYT10*, which indicates that our method is more robust to noisy data. Results on *NYT11* show that neural models (*SPTree*, *Tagging* and *CopyR*) are more effective than pipelined (*FCM*) or feature-based (*MultiR* and *CoType*) methods. *CopyR* is introduced to extract overlapping relations, but it yields poor performance on the *NYT11* test set where there is almost no overlapping relation in a sentence (370 relations among 369 sentences). Whereas our model is still comparable to *SPTree* and performs remarkably better than other baselines. Note that *SPTree* utilizes more linguistic resources (e.g., POS tags, chunks, syntactic parsing trees). This implies that our model is also robust to the data distribution of relations.

Overlapping Relation Extraction

We prepared another two test sets to verify the effectiveness of our model on extracting overlapping relations. Note that overlapping relations can be classified into two types.

- Type I: *two triples share only one entity within a sentence*
- Type II: *two triples share two entities (both head and tail entities) within a sentence*

The first set, *NYT11-plus*, is annotated manually and consists of 149 sentences split from the original *NYT11* training data. The set contains 210/97 overlapping relations for type I/II respectively. The second set, *NYT10-sub*, is a subset of the test set of *NYT10*, and has 715 sentences, but without manual annotation. This set contains 90/2,082 overlapping relations for type I/II respectively. To summarize, most of the overlapping relations in *NYT11-plus* is of type I; while most in *NYT10-sub* is of type II. Table 3 shows the performance of extracting overlapping relations by different approaches.

Results on *NYT10-sub* show that the baselines are very weak to extract overlapping relations of type II on the noisy data, which is consistent with our statement that existing joint extraction approaches cannot deal with overlapping relations effectively in nature. By contrast, our method did not deteriorate too much in performance comparing to that in Table 2, and even obtained a larger gain on *precision*.

Results on *NYT11-plus* demonstrate that our method had a substantial F_1 improvement over all the baselines in extracting overlapping relations of type I on the clean data, indicating that our method can extract overlapping relations more accurately. *SPTree* had a high *precision* but low *recall* since it simply matches one relation type to an entity pair, suffering from ignoring the case of overlapping relations. *Tagging* had low performance in extracting overlapping relations because it assigns a unique tag to an entity even if that entity participates in overlapping relations. Though *CopyR* claimed that it can extract overlapping relations of both types, it fails to extract the relations from clean data effectively as it strongly relies on the annotation of the noisy training data.

To conclude, we can see that extracting overlapping relations is more challenging by comparing results in Table 2 and those in Table 3, and our model is better in extracting two types of overlapping relations no matter the data is noisy or clean.

Interaction between the Two Policies

To justify the effectiveness of integrating entities into a relation and how the interactions are built between the two policies, we investigated the performance on relation detection (classification). In this setting, a prediction is treated as correct as long as the relation type is correctly predicted. The prediction is derived from the high-level policy.

The results in Table 5 demonstrate that our method performs better in relation detection on both datasets. The improvements on *NYT11-plus* are more remarkable as our paradigm is more powerful to extract multiple relations from a sentence. The results indicate that our extraction paradigm which regards entities as arguments of a relation can better capture the relational information in the text.

When removing the low-level entity extraction policy from our model (HRL-Ent), the performance has changed slightly on *NYT11* because each sentence almost contains only one relation in this test set (370 relations among 369 sentences). In this case, the interaction between the two policies has almost no influence on relation detection. However,

[5] As *FCM* cannot detect entity mentions alone, we used the NER results and related features obtained from another baseline *CoType*.

The lawsuit contended that the chairman of the [[News Corporation]Et-Company]Es-Founder , [[[Rupert Murdoch]Es-Company]Et-Founder]Es-Nationality , promised certain rights to shareholders , including the vote on the poison pill , in return for their approval of the company 's plan to reincorporate in the United States from [Australia]Et-Nationality .
Both [Steven A. Ballmer]Es-Company , [[[Microsoft]Et-Company]Et-Company]Es-Founder 's chief executive , and [[Bill Gates]Es-Company]Et-Founder , the chairman , have been involved in that debate inside the company , according to that person .

Table 4: Extraction examples by our model. The words in a bracket represents an entity extracted by the model. *Es* stands for source entity and *Et* for target entity. A predicted relation indicator is marked in background color (e.g. "Murdoch" in the first instance). The entities which form a triple are bracketed in the same color.

Model	NYT11			NYT11-plus		
	Prec	Rec	F_1	Prec	Rec	F_1
FCM	.502	.479	.490	.447	.327	.378
MultiR	.465	.439	.451	.423	.336	.375
CoType	.558	.558	.558	.491	.413	.449
SPTree	.650	.614	.631	**.700**	.343	.460
CopyR	.480	**.714**	.574	.626	.426	.507
HRL-Ent	**.676**	.676	**.676**	.577	.321	.413
HRL	.654	.654	.654	.626	**.456**	**.527**

Table 5: Performance comparison on relation detection.

dramatic drops are observed on *NYT11-plus* where we have 327 relations from 149 sentences, implying that our method (HRL) captures the dependency across multiple extraction tasks and the high-level policy benefits from such interactions. Therefore, our hierarchical extraction framework indeed enhances the interaction between relation detection and entity extraction.

Case Study

Table 4 presents some extraction examples by our model to demonstrate the ability to extract overlapping relations. The first sentence shows the case that an entity pair has multiple relations (type II). Two relations (*Rupert Murdoch*, `person-company`, *News Corporation*) and (*News Corporation*, `company-founder`, *Rupert Murdoch*) share the same entity pair but have different relation types. The model first detects the relation type `person-company` at "Murdoch", and then detects the other relation type `company-founder` at the *comma* position, just next to the word "Murdoch". This shows that relation detection is triggered when sufficient evidence has been gathered at a particular position. And the model can classify the same entities into either source or target entities (for instance, *Rupert Murdoch* is a source entity for `person-company` whereas a target entity for `company-founder`), demonstrating the advantage of our hierarchical framework which can assign dynamic tags to words conditioned on different relation types. In addition, *Rupert Murdoch* has a relation with *Australia*, where the two entities locate far from each other. Though this is more difficult to detect, our model can still extract the relation correctly.

The second sentence gives another example where an entity is involved in multiple relations (type I). In this sentence, (*Steven A. Ballmer*, `person-company`, *Microsoft*) and (*Bill Gates*, `person-company`, *Microsoft*) share the same relation type and target entity, but have different source entities. When the agent scans to the word "Microsoft", the model detects the first relation. The agent then detects the second relation when it scans to the word "Gates". This further demonstrates the benefit of our hierarchical framework which has strengths in extracting overlapping relations by firstly detecting relation and then finding the entity arguments. In addition, our model predicts another relation (*Bill Gates*, `founder-of`, *Microsoft*), which is wrong for this sentence because there is no explicit mention of the relation. This may result from the noise produced by distant supervision, where there are many noisy sentences that are aligned to that relation.

Conclusion and Future Work

In this paper, we present a hierarchical extraction paradigm which approaches relation extraction via hierarchical reinforcement learning. The paradigm treats entities as the arguments of a relation, and decomposes the relation extraction task into a hierarchy of two subtasks: high-level relation indicator detection and low-level entity mention extraction. The high-level policy for relation detection identifies multiple relations in a sentence, and the low-level policy for entity extraction launches a subtask to further extract the related entities for each relation. Thanks to the nature of this hierarchical approach, it is good at modeling the interactions between the two subtasks, and particularly excels at extracting overlapping relations. Experiments demonstrate that our approach outperforms state-of-the-art baselines.

As future work, this hierarchical extraction framework can be generalized to many other pairwise or triple-wise extraction tasks such as aspect-opinion mining or ontology induction.

Acknowledgements

This work was jointly supported by the National Science Foundation of China (Grant No.61876096/61332007), and the National Key R&D Program of China (Grant No. 2018YFC0830200). We would like to thank Prof. Xiaoyan Zhu for her generous support.

Voluptate exercitationem autem at, illum sunt dolorum tempore at nostrum necessitatibus quia ratione obcaecati

cumque error, voluptate cupiditate sit illo, quia nesciunt dolorem asperiores accusamus laudantium magnam, doloremque totam recusandae?