Joint Extraction of Entities and Relations Based on a Novel Decomposition Strategy

Bowen Yu[1,2], Zhenyu Zhang[1,2], Jianlin Su[3], Yubin Wang[1,2], Tingwen Liu[1*], Bin Wang[4], Sujian Li[5]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Zhuiyi Technology Co., Ltd., Shenzhen, China
[4]Xiaomi AI Lab, Xiaomi Inc., Beijing, China
[5]Key Laboratory of Computational Linguistics, Peking University, MOE, China

Abstract

Joint extraction of entities and relations aims to detect entity pairs along with their relations using a single model. Prior works typically solve this task in the extract-then-classify or unified labeling manner. However, these methods either suffer from the redundant entity pairs, or ignore the important inner structure in the process of extracting entities and relations. To address these limitations, in this paper, we first decompose the joint extraction task into two inner-related subtasks, namely HE extraction and TER extraction. The former subtask is to distinguish all head-entities that may be involved with target relations, and the latter is to identify corresponding tail-entities and relations for each extracted head-entity. Next, these two subtasks are further deconstructed into several sequence labeling problems based on our proposed span-based tagging scheme, which are conveniently solved by a hierarchical boundary tagger and a multi-span decoding algorithm. Owing to the reasonable decomposition strategy, our model can fully capture the semantic interdependency between different steps, as well as reduce noise from irrelevant entity pairs. Experimental results show that our method outperforms previous work by 5.6%, 17.2% and 3.7% (F1 score), achieving a new state-of-the-art on three public datasets.

Introduction

Extracting pairs of entities with relations from unstructured text is an essential step in automatic knowledge base construction, and an ideal extraction system should be be capable of extracting overlapping relations (i.e., multiple relations share a common entity) (Zeng et al. 2018). Traditional pipelined approaches first recognize entities, then choose a relation for every possible pair of extracted entities. Such framework makes the task easy to conduct, but ignoring the underlying interactions between these two subtasks (Li and Ji 2014). One improved way is to train them jointly by parameter sharing (Miwa and Bansal 2016; Fu, Li, and Ma 2019; Sun et al. 2019). Although showing promising results, these extract-then-classify approaches still require explicit separate components for entity extraction and relation classification. As a result, their relation classifiers may be misled by the redundant entity pairs (Tan et al. 2019; Dai et al. 2019), since N entities will lead to roughly N^2 pairs, and most of which are in the NA (non-relation) class.

Rather than extracting entities and relations separately, Zheng et al. (2017) propose an unified labeling scheme to model the triplets directly by a kind of multi-part tags. Nevertheless, this model lacks the elegance to identify overlapping relations. As the improvement, Dai et al. (2019) present PA-LSTM which directly labels entities and relations according to query positions, and achieves state-of-the-art results. However, according to our empirical study, this kind of methods always ignore the inner structure such as dependency included in the head entity, tail entity and relation due to the unified labeling-once process. As is well known, a tail-entity and a relation should be depended on a specific head-entity. In other words, if one model does not fully perceive the semantics of head-entity, it will be unreliable to extract the corresponding tail entities and relations. In addition, to recognize overlapping relations, PA-LSTM has to conduct n labeling-once processes for an n-word sentence, which means it is time-consuming and difficult to deploy.

As we see, for a complex NLP task, it is very common to decompose the task into different modules or processes, and a reasonable design is quite crucial to help one model make further progress (Liu et al. 2018; Zhang and Goldwasser 2019; Hu et al. 2019). Thus, in this paper, through analysis of the two kinds of methods above, we exploit the inner structure of joint extraction and propose a novel decomposition strategy, in which the task decomposes hierarchically into several sequence labeling problems with partial labels capturing different aspects of the final task (see Figure 1). Starting with a sentence, we first judiciously distinguish all the candidate head-entities that may be involved with target relations, then label corresponding tail-entities and relations for each extracted head-entity. We call the former subtask as **H**ead-**E**ntity (HE) extraction, and the later as **T**ail-**E**ntity and **R**elation (TER) extraction. Such extract-then-label (ETL) paradigm can be understood by decomposing the joint probability of triplet extraction into conditional probability $p(h, r, t|S) = p(h|S)p(r, t|h, S)$, where (h, r, t) is a triplet in sentence S. In this manner, our TER extractor is able to take the semantic and position information of the given head-entity into account when tagging tail-entities and relations, and naturally, one head-entity can interact with multiple tail-entities to form overlapping relations.

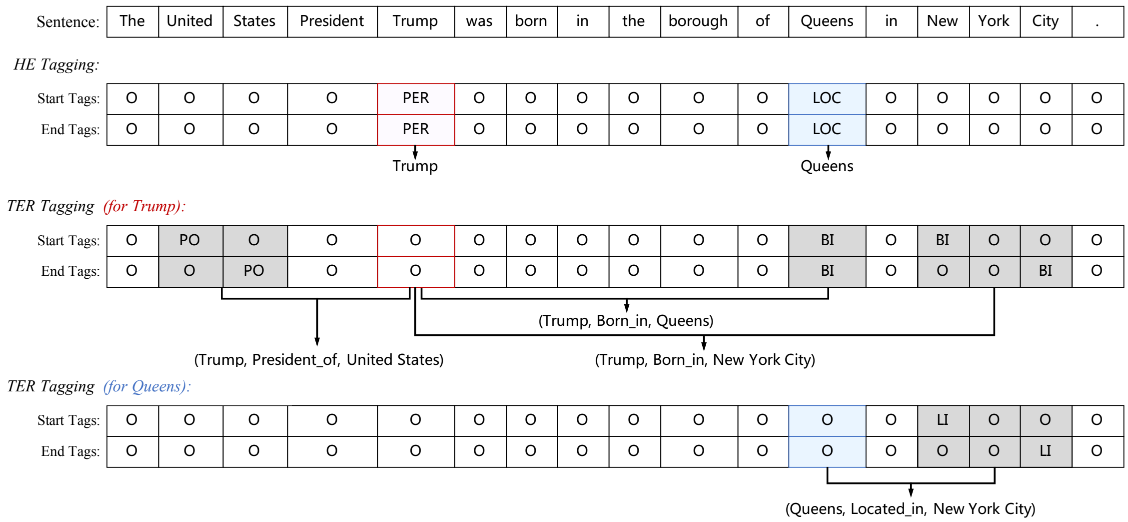

Figure 1: An example of our tagging scheme. *PER* is short for entity type *PERSON*, *LOC* is short for *LOCATION*, *PO* is short for relation type *President_of*, *BI* is short for *Born_in*, and *LI* is short for *Located_in*.

Next, inspired by extractive question answering which identifies answer span by predicting its start and end indices (Seo et al. 2016), we further decompose HE and TER extraction with a span-based tagging scheme. Specifically, for HE extraction, entity type is labeled at the the start and end positions of each head-entity. For TER extraction, we annotate the relation types at the start and end positions of all the tail-entities which have relationship to a given head-entity. To enhance the association between boundary positions, we present a hierarchical boundary tagger, which labels the start and end positions separately in a cascade structure and decode them together by a multi-span decoding algorithm. By this means, HE and TER extraction can be modeled in the unified span-based extraction framework, differentiated only by their prior knowledge and output label set. Overall, for a sentence with m head-entities, the entire task is deconstructed into $2 + 2m$ sequence labeling subtasks, the first 2 for HE tagging and the other $2m$ for TER. Intuitively, the individual subtasks are significantly easy to learn, suggesting that by trained cooperatively with shared underlying representations, they can constrain the learning problem and achieve a better overall outcome.

We conduct experiments on three public datasets: NYT-single, NYT-multi and WebNLG. The results show that our approach significantly outperforms previous work on both normal and overlapping relation extraction, increasing the SOTA F1 score on the three datasets to 59.0% (+5.6), 79.1% (+17.2) and 48.1% (+3.7), respectively.

Methodology

In this section, we first introduce our tagging scheme, based on which the joint extraction task is transformed to several sequence labeling problems. Then we detail the hierarchical boundary tagger, which is the basic labeling module in our method. Finally, we move on to the entire extraction system.

Tagging Scheme

Let us consider the head-entity (HE) extraction first. As discussed in the previous section, it is decomposed into two sequence labeling subtasks. The first sequence labeling subtask mainly focuses on identifying the start position of one head-entity. One token is labeled as the corresponding entity type if it is the start word, otherwise it is assigned the label "*O*" (Outside). In contrast, the second subtask aims to identify the end position of one head-entity and has a similar labeling process except the entity type is labeled for the token which is the end word.

For each identified head-entity, the tail-entity and relation (TER) extractor is also decomposed into two sequence labeling subtasks which make use span boundaries to extract tail-entities and predict relations simultaneously. The first sequence labeling subtask mainly labels the relation type for the token which is the start word of the tail-entity, while the second subtask tags the end word of the tail-entity.

In Figure 1, we illustrate an example to demonstrate our tagging scheme. Based on the scheme, the words "*Trump*", "*United*", "*States*", "*New*", "*City*" and "*Queens*" are all related to the extracted results, thus they are tagged based on our special tags. For example, the word "*Trump*" is the first and also the last word of entity "*Trump*", so the tags are both *PERSON* in the start and end tag sequences when tagging HE. For the TER extraction, when the given head-entity is "*Trump*", there are two tail-entities involved in with a wanted relation, i.e., *("Trump", President_Of, "United States")* and *("Trump", Born_In, "New York City")*, so "*United*" and "*New*" are labeled as *President_Of* and *Born_In* respectively in the start tag sequence. Similarly, we can obtain the end tag sequence that "*States*" and "*City*" are marked. Beyond that, the other words irrelevant to the final result are labeled as "*O*".

Note that our tagging scheme is quite different from PA-LSTM (Dai et al. 2019). For an n-word sentence, PA-LSTM builds n different tag sequences according to different query position while our model tags the same sentence for $2 + 2 \times m$ times to recognize all overlapping relations, where m is the number of head-entities and $m << n$. This means our model is more time-saving and efficient. Besides, it uses "BIES" signs to indicate the position of tokens in the entity while we only predict the start and end positions without loss of the ability to extract multi-word entity mentions.

Hierarchical Boundary Tagger

According to our tagging scheme, we utilize a unified architecture to extract HE and TER. In this paper, we wrap such extractor into a general module named hierarchical boundary tagger (abbreviated as HBT). For the sake of generality, we don't distinguish between head and tail-entity, and they are collectively referred to as targets in this subsection. Formally, the probability of extracting a target t with label l (entity type for head-entity or relation type for tail-entity) from sentence S is modeled as:

$$p(t, l|S) = p(s_t^l|S)p(e_t^l|s_t^l, S) \tag{1}$$

where s_t^l is the start index of t with label l and e_t^l is the end index. Such decomposition indicates that there is a natural order among the tasks: predicting end positions may benefit from the prediction results of start positions, which motivates us to employ a hierarchical tagging structure. As shown in the right panel of Figure 2, we associate each layer with one task and take the tagging results as well as hidden states from the low-level task as input to the high-level. In this work, we choose BiLSTM (Hochreiter and Schmidhuber 1997) as the base encoder. Formally, the label of word x_i when tagging the start position is predicted as Eq. 4.

$$\mathbf{h}_i^{sta} = \mathrm{BiLSTM}_{sta}([\mathbf{h}_i; \mathbf{a}_i]) \tag{2}$$

$$P(y_i^{sta}) = \mathrm{Softmax}(\mathbf{W}^{sta} \cdot \mathbf{h}_i^{sta} + \mathbf{b}^{sta}) \tag{3}$$

$$\text{sta\_tag}(x_i) = \arg\max_k P(y_i^{sta} = k) \tag{4}$$

where $\mathbf{h}_i$ is an input token representation and $\mathbf{a}_i$ is an input auxiliary information vector. When extracting head entities, $\mathbf{a}_i$ is a global representation learned from the entire sentence. It is beneficial to make more accurate predictions from a global perspective. For TER, $\mathbf{a}_i$ is the concatenation of global representation with a head-entity-related vector to indicate the position and semantic information of the given head-entity. Here we adopt BiLSTM_{sta} to fuse $\mathbf{h}_i$ with $\mathbf{a}_i$ into a single vector $\mathbf{h}_i^{sta}$. Analogously, x_i's end tag can be calculated by Eq. 6.

$$\mathbf{h}_i^{end} = \mathrm{BiLSTM}_{end}([\mathbf{h}_i^{sta}; \mathbf{a}_i; \mathbf{p}_i^{se}]) \tag{5}$$

$$P(y_i^{end}) = \mathrm{Softmax}(\mathbf{W}^{end} \cdot \mathbf{h}_i^{end} + \mathbf{b}^{end}) \tag{6}$$

$$\text{end\_tag}(x_i) = \arg\max_k P(y_i^{end} = k) \tag{7}$$

The difference between Eq. 2-4 and Eq. 5-7 is twofold. Firstly, we replace $\mathbf{h}_i$ in Eq. 2 with $\mathbf{h}_i^{sta}$ to make model aware of the hidden states of start positions when predicting end positions. Secondly, inspired by the position encoding vectors used in Zeng et al. (2014), we feed the position embedding $\mathbf{p}_i^{se}$ to the BiLSTM_{end} layer as its additional input. $\mathbf{p}_i^{se}$ can be obtained by looking up p_i^{se} in a trainable position embedding matrix, where

$$p_i^{se} = \begin{cases} i - s^*, & \text{if } s_i^* \text{ exists} \\ C, & \text{otherwise} \end{cases} \tag{8}$$

Algorithm 1 Multi-span decoding

Input:
S, C
S denotes the input sentence
C is a predefined distance constant
Output:
$\{(e_j, tag_j)\}_{j=1}^m$,
e_j denotes the j-th extracted target and tag_j is its type tag
1: Define $n \leftarrow$ Sentence Length
2: Initialize $\mathbf{R} \leftarrow \{\}$
3: Initialize $s^* \leftarrow 0$
4: Initialize p^{se} as a list of length n with default value C
5: Obtain sta_tag(S) by Eq. 4
6: **for** $idx \leftarrow 1$ to n **do**
7:     **if** sta_tag $(S)[idx] \neq$ "O" **then**
8:         $s^* \leftarrow idx$
9:     **if** $s^* > 0$ **then**
10:         $p^{se}[idx] \leftarrow idx - s^*$
11: Obtain $\mathbf{p}^{se}$ by transforming p^{se} into matrix
12: Obtain end_tag(S) by Eq. 7
13: **for** $idx_s \leftarrow 1$ to n **do**
14:     **if** sta_tag$(S)[idx_s] \neq$ "O" **then**
15:         **for** $idx_e \leftarrow idx_s$ to n **do**
16:             **if** end_tag$(S)[idx_e] =$ sta_tag$(S)[idx_s]$ **then**
17:                 $e \leftarrow S[idx_s : idx_e]$
18:                 $tag \leftarrow$end_tag$(S)[idx_e]$
19:                 $\mathbf{R} \leftarrow \mathbf{R} \cup \{(e, tag)\}$
20:                 Break
21: **return R**

Here s^* is the nearest start position before current index, and p_i^{se} is the relative distance between x_i and s^*. When there is no start position before x_i, s^* will not exist, then p_i^s is assigned as a constant C that is normally set to the maximum sentence length. In this way, we explicitly limit the length of the extracted entity and teach model that the end position is impossible to be in front of the start position. To prevent error propagation, we use the gold p^{se} (distance to the correct nearest start position) during training process.

We define the training loss (to be minimized) of HBT as the sum of the negative log probabilities of the true start and end tags by the predicted distributions:

$$\mathcal{L}_{HBT} = -\frac{1}{n}\sum_{i=1}^{n}(\log P(y_i^{sta} = \hat{y}_i^{sta}) + \log P(y_i^{end} = \hat{y}_i^{end})) \tag{9}$$

where $\hat{y}_i^{sta}$ and $\hat{y}_i^{end}$ are the true start and end tags of the i-th word, respectively, and n is the length of the input sentence.

At inference time, to adapt to the multi-target extraction task, we propose a multi-span decoding algorithm, as shown in Algorithm 1. For each input sentence S, we first initialize several variables (Lines 1-4) to assist with the decoding: (1) n is defined as the length of S. (2) $\mathbf{R}$ is initialized as an

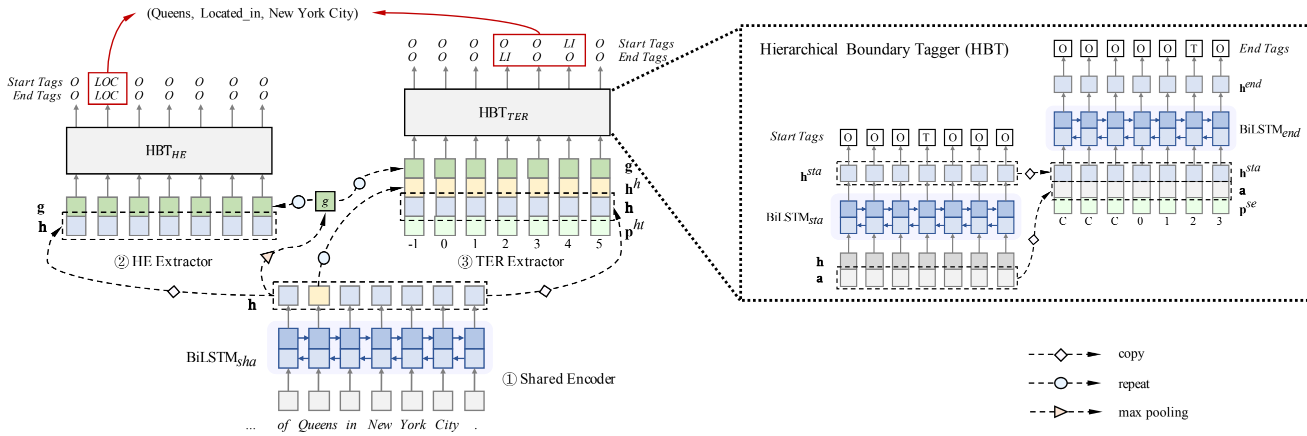

Figure 2: An illustration of our model. The left panel is an overview of our joint extraction system, and the right panel shows the detailed structure of our sequence labeler HBT. Here, "*Queens*" is extracted by the HE extractor, then its hidden state in the shared encoder is marked as the yellow box and entered into the TER extractor as prior knowledge.

empty set to record extracted targets and type tags. (3) s^* is introduced to hold the nearest start position before current index. (4) p^{se} is initialized as a list of length n with default value C to save the position sequence $[p_1^{se}, \cdots, p_n^{se}]$.

Next, we obtain the start tag sequence by Eq. 4 (Line 5) and compute p_i^{se} for each token by Eq. 8 (Lines 6-10). On the basis of p^{se}, we can get $\mathbf{p}^{se}$ by looking up position embedding matrix (Line 11) . Then the tag sequence of end position can be computed by Eq. 7 (Line 12).

Now, all preparations necessary are in place, we start to decoding sta_tag(S) and end_tag(S). We first traverse sta_tag(S) to find the start position of a target (Line 13). If the tag of current index is not "O", it denotes that this position may be a start word (Line 14), then we will traverse end_tag(S) from this index to search for the end position to match the found start position (Line 15). The matching criterion is that if the tag of the end position is identical to the start position (Line 16), the words between the two indices are considered to be a candidate target (Line 17), and the label of start position (or end position) is deemed as the tag of this target (Line 18). The extracted target along with its tag is then added to the set $\mathbf{R}$ (Line 19), and the search in end_tag(S) is terminated to continue to traverse sta_tag(S) to find the next start position (Line 20). Once all the indices in sta_tag(S) are iterated, this decoding function ends by returning the recordset $\mathbf{R}$ (Line 21).

Extraction System

With the span-based tagging scheme and the above hierarchical boundary tagger, we propose an end-to-end neural architecture (Figure 2) to extract entities and overlapping relations jointly. Our model first encodes the n-word sentence using a shared BiLSTM encoder. Then, we build a HE extractor to extract head entities. For each extracted head entity, the TER extractor is triggered with this head-entity's semantic and position information to detect corresponding tail-entities and relations.

Shared Encoder Given sentence $S = \{x_1, \cdots, x_n\}$, we use BiLSTM to incorporate information from both forward and backward directions:

$$\mathbf{h}_i = \mathrm{BiLSTM}_{sha}(\mathbf{x}_i) \tag{10}$$

where $\mathbf{h}_i$ is the hidden state at position i, and $\mathbf{x}_i$ is the word representation of x_i which contains pre-trained embeddings and character-based word representations generated by running a CNN on the character sequence of x_i. We also employ part-of-speech (POS) embedding to enrich $\mathbf{x}_i$.

HE Extractor HE extractor aims to distinguish candidate head-entities and exclude irrelevant ones. We first concatenate $\mathbf{h}_i$ and $\mathbf{g}$ to get the feature vector $\tilde{\mathbf{x}}_i = [\mathbf{h}_i; \mathbf{g}]$, where $\mathbf{g}$ is a global contextual embedding computed by max pooling over all hidden states. Actually, $\mathbf{g}$ works as the $\mathbf{a}_i$ for each token in Eq. 2.

Moreover, we use $\mathbf{H}_{HE} = \{\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_n\}$ to denote all the word representations for HE extraction and subsequently feed $\mathbf{H}_{HE}$ into one HBT to extract head-entities:

$$\mathbf{R}_{HE} = \mathrm{HBT}_{HE}(\mathbf{H}_{HE}) \tag{11}$$

where $\mathbf{R}_{HE} = \{(h_j, type_{h_j})\}_{j=1}^m$ contains all the head-entities and corresponding entity type tags in S.

TER Extractor Similar to HE Extractor, TER Extractor also uses the basic representation $\mathbf{h}_i$ and global vector $\mathbf{g}$ as input features. However, simply concatenating $\mathbf{h}_i$ and $\mathbf{g}$ is not enough for detecting tail-entities and relations with the specific head-entity. The key information required to perform TER extraction includes: (1) the words inside the tail-entity; (2) the depended head-entity; (3) the context that indicates the relationship; (4) the distance between tail-entity and head-entity. Under these considerations, we propose the position-aware, head-entity-aware and context-aware representation $\bar{\mathbf{x}}_i$. Given a head-entity h, we define $\bar{\mathbf{x}}_i$ as follows:

$$\bar{\mathbf{x}}_i = [\mathbf{h}_i; \mathbf{g}; \mathbf{h}^h; \mathbf{p}_i^{ht}] \tag{12}$$

where $\mathbf{h}^h = [\mathbf{h}_{s_h}; \mathbf{h}_{e_h}]$ denotes the representation of head-entity h, in which $\mathbf{h}_{s_h}$ and $\mathbf{h}_{e_h}$ are the hidden states at the start and end indices of h respectively. $\mathbf{p}_i^{ht}$ is the position embedding to encode the the relative distance from current word x_i to h. Obviously, $[\mathbf{g}; \mathbf{h}^h; \mathbf{p}_i^{ht}]$ is the auxiliary feature vector for TER extraction as $\mathbf{a}_i$ in Eq. 2.

It is worth noting that at the training time, h is the gold head-entity, while at the inference time we select head-entity one by one from $\mathbf{R}_{head}$ to complete the extraction task.

Formally, we take $\mathbf{H}_{TER} = \{\bar{\mathbf{x}}_1, \cdots, \bar{\mathbf{x}}_n\}$ as input to one HBT, and the output $\mathbf{R}_{TER} = \{(t_o, rel_o)\}_{o=1}^{z}$, in which t_o is the o-th extracted tail-entity and rel_o is its relation tag with the given head-entity.

$$\mathbf{R}_{TER} = \mathrm{HBT}_{TER}(\mathbf{H}_{TER}) \tag{13}$$

Then we can assemble triplets by combining h and each (t_o, rel_o) to form $\{(h, rel_o, t_o)\}_{o=1}^{z}$, which contains all triplets with head-entity h in sentence S.

Training of Joint Extractor Two learning signals are provided to train the model: $\mathcal{L}_{HE}$ for HE extraction and $\mathcal{L}_{TER}$ for TER extraction, both are formulated as Eq.9. To share input utterance across tasks and train them jointly, for each training instance, we randomly select one head-entity from gold head-entity set as the specified input of the TER extractor. We can also repeat each sentence many times to ensure all triplets are utilized, but the experimental results show that this is not beneficial. Finally, the joint loss is given by:

$$\mathcal{L} = \lambda \mathcal{L}_{HE} + (1 - \lambda) \mathcal{L}_{TER} \tag{14}$$

where λ is a weighting hyperparameter to balance the two components. Then, the model is trained with stochastic gradient descent. Optimizing Eq.14 enables the extraction of head-entity, tail-entity, and relation to be mutually influenced, such that, errors in each component can be constrained by the other.

Experiments

Experiment Settings

Datasets We conduct experiments on three benchmark datasets: (1) NYT-single is sampled from the New York Times corpus (Riedel, Yao, and McCallum 2010) and published by Ren et al. (2017). The training data is automatically labeled using distant supervision, while 395 sentences are annotated manually as test data, most of which have single triplet in each sentence. (2) NYT-multi is published by Zeng et al. (2018) for testing overlapping relation extraction, they selected 5000 sentences from NYT-single as the test set, 5000 sentences as the validation set and the rest 56195 sentences are used as training set. (3) Wiki-KBP is sampled from 780k Wikipedia articles and automatically labeled by Liu et al. (2017), while the test set is selected by Ren et al.(2017). Statistics of the datasets are shown in Table 1.

Evaluation Following previous works, we use the F1 metric computed from Precision (Prec.) and Recall (Rec.) for evaluation. A triplet is marked correct when its relation type and two corresponding entities are all correct. For NYT-single and Wiki-KBP, we create a validation set by randomly sampling 10% sentences from test set as previous studies (Zheng et al. 2017; Dai et al. 2019) did.

Dataset	NYT-single	NYT-multi	WebNLG
# Relation types	24	24	14
# Entity types	3	3	3
# Training sentences	66,335	56,195	75,325
# Test sentences	395	5,000	289

Table 1: Statistics of the datasets.

Implementation Details Following popular choices and previous work, we use the 300 dimension Glove (Pennington, Socher, and Manning 2014) to initialize word embeddings. We randomly initialize the POS, char, and position embeddings with 30-dimension vectors. The window size of CNN for character-based word representations is set to 3, and the number of filters is 50. For the BiLSTM component in our system, we use a 1-layer network with hidden state size 100. Parameter optimization is performed using Adam (Kingma and Ba 2014) with learning rate 0.001 and batch size 64. Dropout is applied to embeddings and hidden states with a rate of 0.4. λ is chosen from $(0, 1)$ via grid search. To prevent the gradient explosion problem, we set gradient clip-norm as 5. All the hyper-parameters are tuned on the validation set. We run 5 times for each experiment, then report the average results.

Comparison Models For comparison, we employ the following models as baselines: (1) **NovelTagging** (Zheng et al. 2017) is the first proposed unified sequence tagger which predicts both entity type and relation class for each word. (2) **MultiDecoder** (Zeng et al. 2018) considers relation extraction as a seq2seq problem and uses dynamic decoders to extract relation triplets. (3) **TME** (Tan et al. 2019) first identifies all candidate entities, then perform relation extraction by ranking candidate relations with the translation mechanism; these two tasks are trained jointly. (4) **PA-LSTM** (Dai et al. 2019) tags entity and relation labels according to a query word position and achieves the recent state-of-the-art result on NYT-single and Wiki-KBP. (5) **GraphRel** (Fu, Li, and Ma 2019) is the latest state-of-the-art method on NYT-multi, which first employs GCNs to extract hidden features, then predicts relations for all word pairs of an entity mention pair extracted by a sequence tagger.

We call our proposed span-based extract-then-label method as **ETL-Span**. In addition, to access the performance influence of our span-based scheme, we also implement another competitive baseline by replacing our tagger with widely used BiLSTM-CRF without any change in the input features ($\tilde{\mathbf{x}}_i$ and $\bar{\mathbf{x}}_i$), and utilize BIES-based scheme accordingly, which associates each type tag (entity type or relation type) with four position tags to indicate the position of entities and types simultaneously, denoted as **ETL-BIES**.

Experimental Results and Analyses

Main Results Table 2 summarizes the comparison results on the three datasets. Overall, our method significantly outperforms others and achieves the state-of-the-art F1 score on

Model	NYT-single			NYT-multi			Wiki-KBP		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
NovelTagging (Zheng et al. 2017)	**61.5**%	41.4%	49.5%	32.8%	30.6%	31.7%	**53.6%**	30.3%	38.7
MultiDecoder (Zeng et al. 2018)	–	–	–	61.0%	56.6%	58.7%	–	–	–
TME (Tan et al. 2019)	50.5%	51.8%	51.1%	–	–	–	–	–	–
PA-LSTM (Dai et al. 2019)	49.4%	59.1%	53.8%	–	–	–	51.1%	39.3%	44.4%
GraphRel (Fu, Li, and Ma 2019)	–	–	–	63.9%	60.0%	61.9%	–	–	–
ETL-BIES	51.1%	64.6%	57.2%	83.4%	72.3%	77.5%	46.1%	48.3%	47.2%
ETL-Span	53.8%	**66.4%**	**59.4%**	**84.1%**	**74.6%**	**79.1%**	46.9%	**49.4%**	**48.1%**

Table 2: Main results on three benchmark datasets. State-of-the-art results are marked in bold. The results for all baselines come directly from the original papers or Dai et al. (2019).

all three datasets. Compared to the current best extrat-then-classify method GraphRel, ETL-Span achieves substantial improvements of 17.2% in F1 on the NYT-multi dataset. We attribute the performance gain to two design choices: (1) the integration of tail-entity and relation extraction as it captures the interdependency between entity recognition and relation classification; (2) the exclusion of redundant (non-relation) entity pairs by the judicious recognition of head-entities which are likely to take part in some relations. For the NYT-single dataset, ETL-Span outperforms PA-LSTM by 5.6% in F1. We consider that it is because (1) we decompose the difficult joint extraction task into several more manageable subtasks and handle them in a mutually enhancing way; and (2) our TER extractor effectively captures the semantic and position information of the depended head-entity, while PA-LSTM detects tail-entities and relations relying on a single query word.

We can also observe that ETL-Span performs remarkably better than ETL-BIES, we guess it is because ETL-BIES must do additional work to learn the semantics of the BIES tags, while in ETL-Span, the entity position is naturally encoded by the set of type labels, thus reducing the tag space of each functional tagger. Another advantage of span-based tagging is that it avoids the computing overhead of CRF, as shown in Table 3, ETL-Span accelerates the decoding speed of ETL-BIES by up to 2.4 times. The main reason is that decoding the best chain of labels with CRF requires a significant amount of computing resources. Besides, ETL-Span only takes about 1/4 time per batch and 1/3 GPU memory compared with ETL-BIES during training, which further verdicts the superiority of our span-based scheme.

We notice that the Precision of our model drops compared with NovelTagging and PA-LSTM on the Wiki-KBP dataset. One possible reason is that many overlapping relations are not annotated in the test data of Wiki-KBP. Following Dai et al. (2019), we add some gold triplets into Wiki-KBP test set and further achieve a large improvement of 13.3% in F1 and 16.9% in Precision compared with the results in Table 2.

Ablation Study To demonstrate the effectiveness of each component, we remove one particular component at a time to understand its impact on the performance. Concretely, we investigated character embedding, POS embedding, Position embedding $\mathbf{p}^{ht}$ and Hierarchical tagging (by tagging

Model	NYT-single	NYT-multi	Wiki-KBP
ETL-BIES	10.9 Bat/s	11.4 Bat/s	16.2 Bat/s
ETL-Span	26.1 Bat/s	25.8 Bat/s	27.9 Bat/s

Table 3: Test-time speed of different models. Bat/s refers to the number of batches can be processed per second.

NYT-single	F_1
ETL-Span	**59.4**
– Char embedding	56.7
– POS embedding	57.6
– Global representation $\mathbf{g}$	56.9
– Position embedding $\mathbf{p}^{ht}$	56.0
– Hierarchical tagging	56.7

Table 4: An ablation study on NYT-single

start positions and end positions at the outmost BiLSTM layer). Table 4 summarizes the results on NYT-single. From these ablations, we find that: (1) Consistent with previous work (Dai et al. 2019), the character-level representations and POS embeddings are helpful to capture the morphological information and deal with OOV words. (2) Introducing global representation $\mathbf{g}$ seems an efficient way to incorporate the information of sentence-level content and make prediction for each word from a global perspective. (3) When we remove $\mathbf{p}^{ht}$, the score drops by 3.4%, which indicates that it is vital to let tail-entity extractor aware of position information of the given head-entity to filter out irrelevant entities by implicit distance constraint. (4) Removing the hierarchical tagging structure hurts the result by 2.7% F1 score, which indicates that predicting end positions benefits from the prediction results of start positions.

Analysis on Joint Learning As shown in Figure 3, we analyze influence of different values of λ on performance of HE, TER and overall triplet extraction. In the process of increasing λ, our model gradually pays more attention to HE extraction and vice versa. It is interesting to see that $\lambda = 1$ leads to the worst HE extraction performance, similar trends are also observable on the TER extraction. This demonstrates that our HE extractor and TER extractor ac-

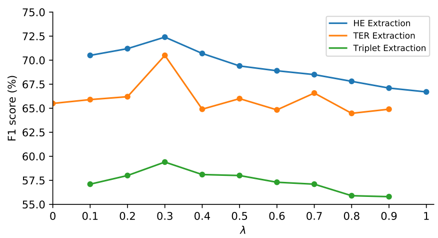

Figure 3: F1 score w.r.t different λ on NYT-single.

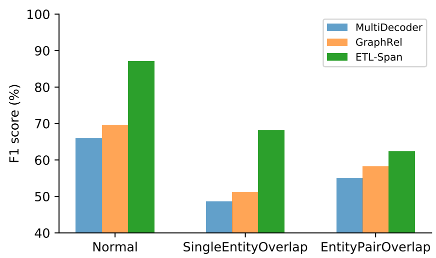

Figure 4: F1 score w.r.t different overlapping categories.

tually work in the mutual promotion way, which again confirms the effectiveness and rationality of our decomposition strategy. Another intriguing observation is that, the performance of all three tasks peaks when $\lambda = 0.3$, which means our model needs to concentrate more on TER extraction, presumably because TER extraction with a larger decision space is more difficult than HE extraction.

Analysis on Overlapping Relation Extraction Following Zeng et al. (2018) and Fu, Li, and Ma (2019), we divide the test set of NYT-multi into three categories: Normal, SingleEntityOverlap (SEO), and EntityPairOverlap (EPO) to verify the effectiveness of our model on extracting overlapping relations. A sentence belongs to Normal class if none of its triplets has overlapping entities. If the entity pairs of two triplets are identical but the relations are different, the sentence will be added to the EPO set. And a sentence belongs to SEO class if some of its triplets have an overlapped entity and these triplets dont have overlapped entity pair. Note that a sentence in the EPO set may contain multiple Normal and SEO triplets. The results are shown in Figure 4[1].

Among the compared baselines, GraphRel and MultiDecoder are the only two models have the capacity to handle the EPO triplets. For this purpose, GraphRel predicts relations for all word pairs, in this case, its relation classifier will be overwhelmed by the superfluous candidates. Readers may have noticed that our model cannot solve the problem of entity pair overlapping. Nevertheless, we still surpass baselines by a substantial margin in all categories. Specifically, our model outperforms GraphRel by 17.4% on the Normal class, 16.9% on the SEO class, and 4.1% on the EPO class. In fact, even on the EPO set, there are still a significant amount of triplets where entity pairs don't overlap. The most common triplets in the real-life corpus are those of Normal and SEO class and our substantial surpass on these two categories masks our shortcomings on the EPO class. We leave the identification of EPO triplets for future work.

[1] Here we don't compare our method with PA-LSTM because PA-LSTM does not release source code, and it is difficult to reproduce the results as in the original papers.

Related Work

Researchers have proposed several methods to extract both entities and relations. Traditional pipelined methods (Zelenko, Aone, and Richardella 2003; Chan and Roth 2011) neglect the relevance of entity extraction and relation prediction. To resolve this problem, several joint models have been proposed. Feature-based works (Yu and Lam 2010; Miwa and Sasaki 2014) need complicated process of feature engineering. Neural models for joint relation extraction are investigated in recent studies (Gupta, Schütze, and Andrassy 2016; Zheng et al. 2017), they show promising results but completely giving up overlapping relations. To overcome this limitation, Zeng et al. (2018) propose a sequence-to-sequence model to decode overlapping relations but fail to generate multi-word entities. Sun et al. (2018) optimize a global loss function to jointly train the two models under the framework work of Minimum Risk Training. Dai et al. (2019) extract triplets by tagging one sentence for n times which is time-consuming with $O(n^2)$ time complexity. TME (Tan et al. 2019) solves this task via ranking with translation mechanism. Takanobu et al. (2019) deal with relation extraction by firstly determining relations and then recognizing entity pairs via reinforcement learning. Li et al. (2019) cast the task as a multi-turn QA problem and generate questions by relation-specific templates. Sun et al. (2019) develop a entity-relation bipartite graph to perform joint inference on entity types and relation types. Fu, Li, and Ma (2019) also utilize graph convolutional network to extract overlapping relations by splitting entity mention pairs into several word pairs and considering all pairs for prediction.

Our span-based tagging scheme is inspired by recent advances in machine reading comprehension (Seo et al. 2016), which derive the answer by predicting its start and the end indices in the paragraph. Hu et al. (2019) also apply this sort of architecture to open-domain aspect extraction. However, unlike these works that predict the start index and end index at one level, our approach passes the prediction information of start indices to higher layer to obtain the end indices, thus better capturing the links between boundary positions.

Conclusion

In this paper, we hierarchically decompose the entity-relation extraction task into several sequence labeling subtasks with partial labels, and solve them in an unified framework. Experimental results show that the functional decomposition of the original task simplifies the learning process and leads to a better overall learning outcome, achieving a new state-of-the-art on three datasets. In the future, we will conduct research on how to apply such decomposition strategy to other information extraction tasks.

References

Chan, Y. S., and Roth, D. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 551–560. Association for Computational Linguistics.

Dai, D.; Xiao, X.; Lyu, Y.; Dou, S.; She, Q.; and Wang, H. 2019. Joint extraction of entities and overlapping relations using position-attentive sequence labeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6300–6308.

Fu, T.-J.; Li, P.-H.; and Ma, W.-Y. 2019. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 1409–1418. Florence, Italy: Association for Computational Linguistics.

Gupta, P.; Schütze, H.; and Andrassy, B. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2537–2547.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hu, M.; Peng, Y.; Huang, Z.; Li, D.; and Lv, Y. 2019. Open-domain targeted sentiment analysis via span-based extraction and classification. *arXiv preprint arXiv:1906.03820*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, Q., and Ji, H. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 402–412.

Li, X.; Yin, F.; Sun, Z.; Li, X.; Yuan, A.; Chai, D.; Zhou, M.; and Li, J. 2019. Entity-relation extraction as multi-turn question answering. *arXiv preprint arXiv:1905.05529*.

Liu, L.; Ren, X.; Zhu, Q.; Zhi, S.; Gui, H.; Ji, H.; and Han, J. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. *arXiv preprint arXiv:1707.00166*.

Liu, L.; Shang, J.; Ren, X.; Xu, F. F.; Gui, H.; Peng, J.; and Han, J. 2018. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Miwa, M., and Bansal, M. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Miwa, M., and Sasaki, Y. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1858–1869.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

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 *Proceedings of the 26th International Conference on World Wide Web*, 1015–1024.

Riedel, S.; Yao, L.; and McCallum, A. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 148–163. Springer.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Sun, C.; Wu, Y.; Lan, M.; Sun, S.; Wang, W.; Lee, K.-C.; and Wu, K. 2018. Extracting entities and relations with joint minimum risk training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2256–2265.

Sun, C.; Gong, Y.; Wu, Y.; Gong, M.; Jiang, D.; Lan, M.; and Sun. 2019. Joint type inference on entities and relations via graph convolutional networks. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 1361–1370.

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

Tan, Z.; Zhao, X.; Wang, W.; and Xiao, W. 2019. Jointly extracting multiple triplets with multilayer translation constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yu, X., and Lam, W. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, 1399–1407. Association for Computational Linguistics.

Zelenko, D.; Aone, C.; and Richardella, A. 2003. Kernel methods for relation extraction. *Journal of machine learning research* 3(Feb):1083–1106.

Zeng, D.; Liu, K.; Lai, S.; Zhou, G.; and Zhao, J. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2335–2344.

Zeng, X.; Zeng, D.; He, S.; Liu, K.; and Zhao, J. 2018. 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)*, 506–514.

Zhang, X., and Goldwasser, D. 2019. Sentiment tagging with partial labels using modular architectures. *arXiv preprint arXiv:1906.00534*.

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 *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1227–1236.