Improved Decomposition Strategy for Joint Entity and Relation Extraction

Van-Hien Tran[†], Van-Thuy Phi[†], Akihiko Kato[†], Hiroyuki Shindo[†],
Taro Watanabe[†] and Yuji Matsumoto^{††}

The joint entity and relation extraction task detects entity pairs along with their relations to extract relational triplets. A recent study (Yu et al. 2020) proposed a novel decomposition strategy that splits the task into two interrelated subtasks: detection of the head-entity (*HE*) and identification of the corresponding tail-entity and relation (*TER*) for each extracted head-entity. However, this strategy suffers from two major problems. First, if the *HE* detection task fails to find a valid head-entity, the model will then miss all related triplets containing this head-entity in the *head* role. Second, as Yu et al. (2020) stated, their model cannot solve the entity pair overlap (EPO) problem. For a given head-entity, the *TER* extraction task predicts only a single relation between the head-entity and a tail-entity, even though this entity pair can hold multiple relations. To address these problems, we propose an improved decomposition strategy that considers each extracted entity in two roles (*head* and *tail*) and allows a model to predict multiple relations (if any) of an entity pair. In addition, a corresponding model framework is presented to deploy our new decomposition strategy. Experimental results showed that our approach significantly outperformed the previous approach of Yu et al. (2020) and achieved state-of-the-art performance on two benchmark datasets.

Key Words: *Information Extraction, Joint Extraction, Named-Entity Recognition, Relation Classification, Decomposition Strategy*

1 Introduction

The extraction of relational triplets is a critical and challenging task in natural language processing (NLP). Given an unstructured text, it aims to extract pairs of entities with semantic relations, in the form of (*head*, *relation*, *tail*). The relational triplets extraction has attracted considerable research effort as it plays a vital role in many NLP applications such as information extraction (Tran et al. 2021) and question answering (Hao et al. 2017). For example, in information extraction, given a biomedical text, it is expected to extract both the biomedical entities and

[†] Nara Institute of Science and Technology, Japan

^{††} RIKEN Center for Advanced Intelligence Project (AIP)

their relations in the form of triplets such as (“*coronavirus*”, “causes”, “*respiratory infections*”) and (“*tocilizumab*”, “treats”, “*cytokine release syndrome*”).

Traditional pipeline works (Zelenko et al. 2003; Zhou et al. 2005; Chan and Roth 2011) divide this task into two isolated subtasks: named-entity recognition (NER) (Vu et al. 2015) and relation classification (RC) (Tran et al. 2019). Specifically, they first recognize all the entities and then predict relations between the extracted entities. Such methods tend to suffer from error propagation and ignore the relevance between the two subtasks. To address these problems, subsequent studies proposed joint learning of entities and relations in a single model, including feature-based models (Yu and Lam 2010; Li and Ji 2014; Ren et al. 2017) and neural network-based models (Gupta et al. 2016; Katiyar and Cardie 2017; Zeng et al. 2018; Fu et al. 2019; Yu et al. 2020).

One of the biggest challenges of this task is the *overlapping triplet problem*, which is expressed in two scenarios: *entity pair overlap (EPO)* and *single entity overlap (SEO)*. Specifically, EPO occurs when triplets share the same entity pair but with different relations, such as: (“*Paris*”, “Capital_of”, “*France*”), (“*Paris*”, “Located_in”, “*France*”), and (“*Paris*”, “Administrative_division_of”, “*France*”), as shown in Figure 1. SEO occurs when two relational triplets share only one common entity, such as: (“*John Smiths*”, “Work_in”, “*Paris*”) and (“*John Smiths*”, “Live_in”, “*France*”).

Most previous works could not efficiently address the *overlapping triplet problem*. This problem directly challenges conventional sequence tagging schemes, in which each token represents only a single tag (Zheng et al. 2017). It also creates significant difficulties in traditional RC approaches, where an entity pair is supposed to hold at most one relation (Miwa and Bansal 2016). Zeng et al. (2018) is among the first to solve the problem by proposing a sequence-to-sequence model with a copy mechanism. Fu et al. (2019) utilized a graph convolutional network to extract

Fig. 1 An example of the *entity pair overlap (EPO)* and *single entity overlap (SEO)* triplets.

overlapping triplets. In contrast to the previous works, Yu et al. (2020) presented a unified sequence labeling framework based on a novel decomposition strategy. However, this method can only deal with the SEO triplets in the sample and fails to handle the EPO cases, as Yu et al. (2020) stated.

Specifically, Yu et al. (2020) decomposed the joint task into two subtasks: head-entity extraction and tail-entity relation extraction. The first task detects all head-entities, whereas the second one detects the corresponding tail-entities and target relations for a given head-entity. Although this method significantly outperforms previous methods, it suffers from two issues. **First**, to create relational triplets, it always detects head-entities first and then extracts the corresponding tail-entities and relations for each detected head entity. Thus, observably, if the first task fails to find a valid head-entity, the model will then miss all the related triplets containing this head-entity in the *head* role. **Second**, as Yu et al. (2020) stated, their model cannot solve the *overlapping triplet problem* in the EPO scenario. For a given head-entity, the second task predicts only a single relation between the given head-entity and any corresponding tail-entity, even though this entity pair can hold multiple relations.

Therefore, we propose an improved decomposition strategy to overcome these two problems. For the **first issue**, we designed a more flexible strategy. We detect all entities first, and then, for each extracted entity, we identify it in each (*head* / *tail*) entity role and extract the corresponding (*tail-entities* / *head-entities*) and relations. For the **second issue**, we define a set of “*unified relation labels*” (*URLs*), each of which represents a unique (unordered) subset of the full set of original relations. By using these *URLs* in a multiclass classifier, our model can solve the EPO problem. In addition, a corresponding model framework is introduced to deploy our new strategy. The experimental results on both two benchmark datasets showed that our approach significantly outperformed the previous approach of Yu et al. (2020) as well as previous state-of-the-art approaches.

2 Methodology

In this section, we first introduce the decomposition strategy of Yu et al. (2020) and then present our new strategy. In addition, a corresponding model framework is proposed for deploying our decomposition strategy.

2.1 Tagging Scheme

Yu et al. (2020) decomposed the joint extraction task into two interrelated subtasks: *Head-Entity* (*HE*) extraction and *Tail-Entity Relation* (*TER*) extraction. The *HE* extraction task is modeled by two sequence labeling tasks, one for identifying the start position and the other for the end position of the head-entities, respectively. The entity type is also labeled simultaneously at the head-entity positions. Meanwhile, for each identified head-entity, the *TER* extraction task is also modeled by two sequence labeling tasks, one for detecting the start position and the other for detecting the end position of the corresponding tail-entities. As is done for the *HE* detection, the relation type between the given head-entity and its corresponding tail-entity is also labeled in each position.

Figure 2(a) illustrates an example of the above tagging scheme. From the input sample, the *HE* tagging detects the HEs: “*John Smiths*” and “*Paris*” because they are HEs in the set of gold triplets. Then, for the given HE “*Paris*”, the *TER* tagging identifies the tail-entity “*France*”

Fig. 2 (a) Tagging scheme of Yu et al. (2020). (b) Our tagging scheme. *PER* and *LOC* stand for *PERSON* and *LOCATION*, respectively. *HE*, *TER*, and *HTER* stand for *Head-Entity*, *Tail-Entity Relation*, and *Head/Tail Entity Relation*, respectively. The set of gold triplets is: {(“*John Smiths*”, R_1 , “*Paris*”), (“*John Smiths*”, R_2 , “*Paris*”), (“*John Smiths*”, R_1 , “*France*”), (“*John Smiths*”, R_2 , “*France*”), (“*Paris*”, R_3 , “*France*”), (“*Paris*”, R_4 , “*France*”), (“*Paris*”, R_5 , “*France*”)}, where R_1 : “*Live_in*”; R_2 : “*Work_in*”; R_3 : “*Capital_of*”; R_4 : “*Located_in*”; and R_5 : “*Administrative_division_of*”. In (b), $\hat{R}_1$ and $\hat{R}_2$ are URLs, where $\hat{R}_1$: { R_1, R_2 } and $\hat{R}_2$: { R_3, R_4, R_5 }.

with the expected relation R_3 because of the gold triplet: (“Paris” , R_3 , “France”). However, this tagging scheme suffers from two existing problems (as mentioned in Section 1) that hinder a further improvement of the system performance. We explain in detail how our new decomposition strategy can solve these two issues.

First, to obtain relational triplets in the form of (*head*, *relation*, *tail*), the model by Yu et al. (2020) always detects the HEs first and then extracts the corresponding tail-entities and relations for each detected HE. Following this strategy, if *HE* tagging fails to find a valid HE, the model will then miss all the related triplets. For instance, in Figure 2(a), if “Paris” is not identified as a HE, the model will miss all gold triplets containing “Paris” in the *head* role. Meanwhile, it is not always easy to extract head entities first for all relations, especially when the relation types are diverse. To deal with this issue, we designed a new strategy, which is illustrated in Figure 2(b). This strategy allows our model not only to learn the probability distribution closer to the gold labels but also to increase the chances of extracting a valid triplet, which may be overlooked by the approach of Yu et al. (2020). Specifically, we first extract all entities without differentiating the *head/tail* role using the *Entity* tagging in our scheme. For each extracted entity, the *head/tail entity relation* (*HTER*) tagging considers it in each *head/tail* role and detects all corresponding *tail entities/head entities* and relation types, respectively. For example, in Figure 2(b), the *Entity* tagging detects the entities: “John Smiths” , “Paris” , and “France” . Then, for the given entity “Paris” , the *HTER* tagging considers it in the *head* role to identify the tail-entity “France” with the *unified relation label* (URL) $\hat{R}_2$, and also considers “Paris” in the *tail* role to recognize the HE “John Smiths” with the URL $\hat{R}_1$.

Second, the previous tagging scheme cannot solve the EPO problem. For instance, in Figure 2(a), the entity pair (“Paris” , “France”) holds multiple relations: R_3 , R_4 , and R_5 . However, for the given HE “Paris” , the *TER* tagging can predict only one of the original relations to the tail-entity “France” , using a multi-class classifier of (N_R+1) classes, which include N_R original relations and one special class *No_relation*. To overcome this limitation, we propose two different solutions. First, in a natural way, we use a multi-label classifier to detect multiple relations (if any) between an ordered entity pair. With this solution, each tagging position in the *HTER* tagging can hold multiple original relation types (if any), instead of only a maximum of a single relation type (if any), as assumed by Yu et al. (2020). However, in practice, the maximum number

of relation types co-occurring between an entity pair is often small¹. For instance, the maximum number of relation types for any entity pair is only 3 in both the NYT (Riedel et al. 2010) and the WebNLG (Gardent et al. 2017) datasets, while the total number of original relations on the NYT and WebNLG datasets is 24 and 216, respectively. Consequently, the sparse label problem on the relation types of the same entity pair can affect the system performance, especially in the WebNLG dataset. Therefore, we propose a second solution that uses a multi-class classifier with a set of *URLs* to deal with both the sparse label problem and the EPO problem. In essence, the purpose of using the created *URLs* is to transform the “multi-label classification task with a sparse label problem on the set of original relations” into the “multi-class classification task on the set *URLs*”.

Using the training set $\mathbf{D}$ and a predefined threshold γ , following Algorithm 1, we create the set *URLs*. Specifically, first of all, for each ordered entity pair $\mathbf{p}$ in each sample in $\mathbf{D}$, the function $F(\mathbf{p})$ returns a single *URL* $\hat{R}$ that represents a unique (unordered) subset, where this

Algorithm 1 Creation of a set of “*unified relation labels*”

Input: $\mathbf{D}$: training dataset; γ : a pre-defined threshold.

Output: *URLs*, the expected “*unified relation labels*” set.

```

1: Initialize an empty dictionary:  $Q \leftarrow \{\}$ 
2: for each sample  $\mathcal{X}$  in  $\mathbf{D}$  do
3:   for each ordered entity pair  $\mathbf{p}$  in  $\mathcal{X}$  do
4:      $\hat{R} = F(\mathbf{p})$ 
5:     if  $\hat{R} \neq \emptyset$  and  $\hat{R}$  not in  $Q$  then
6:        $Q[\hat{R}] = 0$ 
7:     end if
8:      $Q[\hat{R}] = Q[\hat{R}] + 1$ 
9:   end for
10: end for
11: for each  $\hat{R}$  in  $Q$  do
12:   if  $Q[\hat{R}] \geq \gamma$  then
13:     Add  $\hat{R}$  to the set URLs
14:   end if
15: end for
16: return URLs

```

¹ There are two possible reasons for this phenomenon. First, because each relation type is usually associated with certain entity types (e.g., “*Live.in*” is between *Person* and *Location*), relation types co-occurring in the same entity pair are often required to share the same entity type pair. Second, in the joint extraction task, as each entity is often mentioned only once and the length of the input sample is not very long, they lead to the limited expressions of the possible relations of the same entity pair.

subset includes all the existing original relations of the pair $\mathbf{p}$. We then count the frequency of each $\hat{R}$ on the entire $\mathbf{D}$ and only keep $\hat{R}$ when its frequency is greater than or equal to γ . With the obtained *URLs*, for each ordered entity pair (*head*, *tail*) in each sample in $\mathbf{D}$, we replace the full set of all existing original relations of this entity pair with a single corresponding *URL* in the set *URLs*. Conversely, we ignore relation sets if they do not match any corresponding *URLs* in the set *URLs*. Note that we only performed this label conversion for the training set, but not for the validation and test sets. With this procedure, any ordered entity pairs in any sentences in the training dataset $\mathbf{D}$ will now have only a single *URL* in the set *URLs* or have no relation. Finally, we train the model on the training dataset $\mathbf{D}$ with the *URLs* instead of with the set of original relations.

In Table 1, we provide a toy example for creating *URLs* using Algorithm 1 and for using them on the training set $\mathbf{D}$. Assume that the training set $\mathbf{D}$ includes two samples, where each sample has its gold relational triplets. By using Algorithm 1, we obtain the dictionary $\mathbf{Q}$, which contains all the “*URLs*” along with their frequencies. With the predefined threshold γ (e.g.; $\gamma = 1$), we obtain the set *URLs*: $\{\hat{R}_1, \hat{R}_2, \hat{R}_3, \hat{R}_4\}$. Then, using the created set *URLs*, for each entity pair in each sample, we replace all existing original relations of this pair with a single corresponding *URL* in the set *URLs* (if any). For instance, in Sample 2, the ordered entity pair: (“*Alex*”, “*Spain*”) with the original relations: {“*Work_in*”, “*Place_of_birth*”, “*Place_of_death*”} will become: (“*Alex*”, $\hat{R}_4$, “*Spain*”). Finally, our designed model will be trained on the training set $\mathbf{D}$ with the set *URLs*.

$\mathbf{D}$	Sample 1	Harry works as an artist in Rome, the capital of Italy. (“ <i>Harry</i> ”, “ <i>Occupation</i> ”, “ <i>artist</i> ”), (“ <i>Harry</i> ”, “ <i>Work_in</i> ”, “ <i>Rome</i> ”), (“ <i>Harry</i> ”, “ <i>Work_in</i> ”, “ <i>Italy</i> ”), (“ <i>Rome</i> ”, “ <i>Capital_of</i> ”, “ <i>Italy</i> ”), (“ <i>Rome</i> ”, “ <i>Located_in</i> ”, “ <i>Italy</i> ”)
	Sample 2	Alex, a talented writer, was born and passed away in Spain, where he worked all his life. (“ <i>Alex</i> ”, “ <i>Occupation</i> ”, “ <i>writer</i> ”), (“ <i>Alex</i> ”, “ <i>Place_of_birth</i> ”, “ <i>Spain</i> ”), (“ <i>Alex</i> ”, “ <i>Place_of_death</i> ”, “ <i>Spain</i> ”), (“ <i>Alex</i> ”, “ <i>Work_in</i> ”, “ <i>Spain</i> ”)
Original Relations		“ <i>Occupation</i> ”, “ <i>Work_in</i> ”, “ <i>Capital_of</i> ”, “ <i>Located_in</i> ”, “ <i>Place_of_birth</i> ”, “ <i>Place_of_death</i> ”
Unified Relation Labels		$\hat{R}_1$: {“ <i>Occupation</i> ”}, $\hat{R}_2$: {“ <i>Work_in</i> ”}, $\hat{R}_3$: {“ <i>Capital_of</i> ”, “ <i>Located_in</i> ”}, $\hat{R}_4$: {“ <i>Place_of_birth</i> ”, “ <i>Place_of_death</i> ”, “ <i>Work_in</i> ”}.
Dict $\mathbf{Q}$ and the set <i>URLs</i>		$\mathbf{Q} = \{\hat{R}_1: 2, \hat{R}_2: 2, \hat{R}_3: 1, \hat{R}_4: 1\}$. <i>URLs</i> = $\{\hat{R}_1, \hat{R}_2, \hat{R}_3, \hat{R}_4\}$ when $\gamma = 1$.
$\mathbf{D}$ with the set <i>URLs</i>	Sample 1	Harry works as an artist in Rome, the capital of Italy. (“ <i>Harry</i> ”, $\hat{R}_1$, “ <i>artist</i> ”), (“ <i>Harry</i> ”, $\hat{R}_2$, “ <i>Rome</i> ”), (“ <i>Harry</i> ”, $\hat{R}_2$, “ <i>Italy</i> ”), (“ <i>Rome</i> ”, $\hat{R}_3$, “ <i>Italy</i> ”)
	Sample 2	Alex, a talented writer, was born and passed away in Spain, where he worked all his life. (“ <i>Alex</i> ”, $\hat{R}_1$, “ <i>writer</i> ”), (“ <i>Alex</i> ”, $\hat{R}_4$, “ <i>Spain</i> ”)

Table 1 A toy example of creating and using the set *URLs* on the training set $\mathbf{D}$.

2.2 Network Structure

Following our tagging scheme in Figure 2(b), we present our corresponding model framework in Figure 3. It consists of three main parts: *Encoding Layer*, *Entity Extractor*, and *HTER Extractor*.

2.2.1 Encoding Layer

Given a sample $X = \{x_1, x_2, \dots, x_N\}$ with N tokens, we first utilize a bidirectional long short-term memory (BiLSTM) (Hochreiter and Schmidhuber 1997) network to encode the contextualized representation for each token. The initial embedding $\mathbf{e}_i$ of each input token is concatenated by three parts: pre-trained word embedding, character-level word embedding generated by a convolutional neural network (CNN) on the character sequence of x_i , and a part-of-speech (POS) embedding. Then, the contextualized representation sequence $H = \{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_N\}$ is obtained as follows:

$$\mathbf{h}_i = [\vec{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i], \quad (1)$$

Fig. 3 Our framework. We used the same input sample as in Figures 1 and 2. Here, the extracted entity “Paris” is entered into the HTER Extractor as prior knowledge. In the HTER Extractor, $\hat{R}_1$ and $\hat{R}_2$ are in the set $URLs$ created using Algorithm 1, where $\hat{R}_1$: {“Live_in”, “Work_in”}, $\hat{R}_2$: {“Capital_of”, “Located_in”, “Administrative_division_of”}. Note that the HTER Extractor was trained with the set $URLs$, instead of with the set of original relations.

$$\vec{\mathbf{h}}_i = \text{LSTM}^f(\mathbf{e}_i, \vec{\mathbf{h}}_{i-1}), \quad \overleftarrow{\mathbf{h}}_i = \text{LSTM}^b(\mathbf{e}_i, \overleftarrow{\mathbf{h}}_{i+1}), \quad (2)$$

where LSTM^f and LSTM^b denote the forward and backward LSTM, respectively.

2.2.2 Entity Extractor

The Entity Extractor module aims to recognize the relevant entities in the input sample by directly decoding the output sequence H of the Encoding Layer. Specifically, it adopts two identical multiclass classifiers to detect the start and end positions of the entities with the corresponding entity type label. Formally, the detailed operations of the entity tagging on each token are as follows:

$$p_i^{\text{start-ent}} = \text{Softmax}(\mathbf{W}_{\text{start-ent}}\mathbf{h}_i + \mathbf{b}_{\text{start-ent}}), \quad (3)$$

$$p_i^{\text{end-ent}} = \text{Softmax}(\mathbf{W}_{\text{end-ent}}\mathbf{h}_i + \mathbf{b}_{\text{end-ent}}), \quad (4)$$

where $p_i^{\text{start-ent}}$ and $p_i^{\text{end-ent}}$ represent the probabilities of the entity type labels for the i^{th} token, which are considered as the start and end positions of an entity, respectively. In addition, $\mathbf{h}_i$ is the encoded representation, $\mathbf{W}_{(\cdot)}$ represents the trainable weight, and $\mathbf{b}_{(\cdot)}$ is the bias.

We define the training loss (to be minimized) of the Entity Extractor as the sum of the negative log probabilities of the true *start* and *end* tags, using the predicted distributions:

$$\mathcal{L}_E = -\frac{1}{N} \sum_{i=1}^N \left(\log P(y_i^{\text{start-ent}} = \hat{y}_i^{\text{start-ent}}) + \log P(y_i^{\text{end-ent}} = \hat{y}_i^{\text{end-ent}}) \right), \quad (5)$$

where $\hat{y}_i^{\text{start-ent}}$ and $\hat{y}_i^{\text{end-ent}}$ are the true *start* and *end* tags (gold labels) of the i^{th} word in the sample X , respectively, and N is the length of the sample X .

2.2.3 HTER Extractor

The *HTER Extractor* consists of two submodules: **Head-Entity Relation (HER)** extractor and **Tail-Entity Relation (TER)** extractor. For each given entity, e.g., “Paris”, it uses the *TER* to identify “Paris” in the *head* entity role and detect all the corresponding tail-entities and *URLs*, such as (“**Paris**”, $\hat{R}_2$, “*France*”), where $\hat{R}_2: \{ \text{“Capital_of”, “Located_in”, “Administrative_division_of”} \}$. At the same time, the HTER Extractor utilizes the *HER* submodule to identify “Paris” in the *tail* entity role and detect all the corresponding head-entities and *URLs*, such as (“*John Smiths*”, $\hat{R}_1$, “**Paris**”), where $\hat{R}_1: \{ \text{“Live_in”, “Work_in”} \}$.

Specifically, from the output sequence H of the Encoding Layer, as an entity is often composed of multiple tokens, we create a span feature representation for the given entity. Following Ouchi

et al. (2018), for the entity with start and end positions: j and k ($j \leq k$), we obtain the entity representation vector as follows:

$$\mathbf{v}_{ent} = [\mathbf{h}_j + \mathbf{h}_k; \mathbf{h}_j - \mathbf{h}_k], \quad (6)$$

$$\bar{\mathbf{x}}_i = [\mathbf{h}_i; \mathbf{v}_{ent}], \quad (7)$$

where i refers to the position of the i^{th} word in the input sample.

Because the information of a given entity is crucial for extracting related triplets, we therefore concatenate each token vector in the output sequence H and the given entity representation $\mathbf{v}_{ent}$. We take $\bar{\mathbf{X}} = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \dots, \bar{\mathbf{x}}_N\}$ as the input to another BiLSTM layer, to fuse each $\mathbf{h}_i$ and $\mathbf{v}_{ent}$ in a single vector $\bar{\mathbf{h}}_i$:

$$\bar{\mathbf{H}} = BiLSTM(\bar{\mathbf{X}}), \quad (8)$$

where $\bar{\mathbf{H}} = \{\bar{\mathbf{h}}_1, \bar{\mathbf{h}}_2, \dots, \bar{\mathbf{h}}_N\}$. Then, the sequence $\bar{\mathbf{H}}$ is used as the same input to both *TER* and *HER* submodules. The *TER* submodule detects all the corresponding tail-entities and relations by directly decoding the sequence $\bar{\mathbf{H}}$. Specifically, it uses two identical multiclass classifiers to detect the start and end positions of the related tail-entities with the corresponding relation type. Thus, the detailed operations of the *tail* entity tagging with the relation type on each token are described as follows:

$$p_i^{start_tail} = Softmax(\mathbf{W}_{start_tail}\bar{\mathbf{h}}_i + \mathbf{b}_{start_tail}), \quad (9)$$

$$p_i^{end_tail} = Softmax(\mathbf{W}_{end_tail}\bar{\mathbf{h}}_i + \mathbf{b}_{end_tail}), \quad (10)$$

where $p_i^{start_tail}$ and $p_i^{end_tail}$ represent the probabilities of the relation labels for the i^{th} token, which are considered as the start and end positions of a *tail* entity in the input sample, respectively. Additionally, $\bar{\mathbf{h}}_i$ is the encoded representation, $\mathbf{W}_{(.)}$ represents the trainable weight, and $\mathbf{b}_{(.)}$ is the bias.

Similarly, the *HER* submodule utilizes two other identical multi-class classifiers to detect the start and end positions of the related head-entities with the corresponding relation type. Formally, the detailed operations of the *head* tagging on each token are as follows:

$$p_i^{start_head} = Softmax(\mathbf{W}_{start_head}\bar{\mathbf{h}}_i + \mathbf{b}_{start_head}), \quad (11)$$

$$p_i^{end_head} = Softmax(\mathbf{W}_{end_head}\bar{\mathbf{h}}_i + \mathbf{b}_{end_head}). \quad (12)$$

Therefore, we have the loss function of each submodule in the *HTER Extractor* as follows:

$$\mathcal{L}_{TER} = -\frac{1}{N} \sum_{i=1}^N \left(\log P \left(y_i^{start_tail} = \hat{y}_i^{start_tail} \right) + \log P \left(y_i^{end_tail} = \hat{y}_i^{end_tail} \right) \right), \quad (13)$$

$$\mathcal{L}_{HER} = -\frac{1}{N} \sum_{i=1}^N \left(\log P \left(y_i^{start_head} = \hat{y}_i^{start_head} \right) + \log P \left(y_i^{end_head} = \hat{y}_i^{end_head} \right) \right), \quad (14)$$

where N is the length of the input sample; $\hat{y}_i^{start_tail}$ and $\hat{y}_i^{end_tail}$ in Eq. 13 are the true *start* and *end* relation tags of the i^{th} word for annotating the related tail entities, respectively, and $\hat{y}_i^{start_head}$ and $\hat{y}_i^{end_head}$ in Eq. 14 are the true *start* and *end* relation tags of the i^{th} word for annotating the related head entities.

2.2.4 Joint Learning

To boost the interaction between the *Entity Extractor* and the *HTER Extractor*, we combine their loss functions to form the entire loss objective of our model:

$$\mathcal{L}(\theta) = \alpha * \mathcal{L}_E + (\mathcal{L}_{TER} + \mathcal{L}_{HER}), \quad (15)$$

where the hyper-parameter α is fine-tuned in the range $(0, 1]$. Then, we train the model by minimizing $\mathcal{L}(\theta)$ through the Adam stochastic gradient descent (Kingma and Ba 2014) over shuffled mini-batches. Note that the *HTER Extractor* is trained with the gold set *URLs*, which are created using Algorithm 1, instead of with the set of original relations.

2.2.5 Inference

In the testing phase, the triplets can be easily inferred on the basis of the two modules. Specifically, for each input sample, we first extract the entities by using the *Entity Extractor* module. Note that entities extracted by this module will not be considered as an additional constraint on the output of the other module. Then, for each detected entity, we utilize the *HTER Extractor* to consider it in the *head/tail* roles and extract all the relational triplets involving this entity. For example, from the input sample in Figure 3, the *Entity Extractor* is expected to detect the entities: “John Smiths”, “Paris”, and “France”. Then, for each extracted entity, e.g., “Paris”, the *HTER Extractor* uses its two submodules (*HER* and *TER*) to extract all relational triplets containing “Paris”. Specifically, the *TER* submodule identifies “Paris” in the *head* role and extracts: (“**Paris**”, $\hat{R}_2$, “France”). Meanwhile, the *HER* submodule considers “Paris” in the *tail* role and extracts: (“John Smiths”, $\hat{R}_1$, “**Paris**”).

Note that the relation types in the triplets extracted by both the *HER* and *TER* submodules

belong to the set *URLs* because they are trained with this set. Thus, we need to transform these relations into the original relations by breaking them down into the original relations and creating the corresponding triplets. In the above example, for the given entity “*Paris*”, the *TER* submodule extracts the triplets $\{(\text{“Paris”}, \hat{R}_2, \text{“France”})\}$. In addition, as shown in Figure 2, $\hat{R}_2$ represents for the subset $\{\text{“Capital_of”}, \text{“Located_in”}, \text{“Administrative_division_of”}\}$. Therefore, we obtain the final triplets from the *TER* submodule for “*Paris*”: $\{(\text{“Paris”}, \text{“Capital_of”}, \text{“France”}), (\text{“Paris”}, \text{“Located_in”}, \text{“France”}), (\text{“Paris”}, \text{“Administrative_division_of”}, \text{“France”})\}$. Similarly, we also obtain the triplets from the *HER* submodule for “*Paris*”: $\{(\text{“John Smiths”}, \text{“Live_in”}, \text{“Paris”}), (\text{“John Smiths”}, \text{“Work_in”}, \text{“Paris”})\}$. Finally, we combine the outputs from both submodules by keeping all the extracted triplets, but removing the duplicates (if any) for each input sample.

3 Experiments

3.1 Experimental Settings

3.1.1 Datasets and Evaluation Metrics

Following the previous work (Dai et al. 2019; Yu et al. 2020), we evaluated our approach on two widely used datasets: **NYT** (Riedel et al. 2010) and **WebNLG** (Gardent et al. 2017). To further study the capability of our approach to extract overlapping and multiple relations, we also split the test set into three categories: *Normal*, *EPO*, and *SEO*. A sample belongs to *Normal* if none of its triplets overlaps, whereas it belongs to *EPO* if some of its triplets share the same entity pair. In addition, a sample belongs to *SEO* if some of its triplets share only one common entity. The statistics of the two datasets are given in Table 2.

We report the standard micro precision, recall, and F1-score, as in line with recent studies.

Dataset	Train	Valid	Test	Category			No. of Relations
				Normal	SEO	EPO	
NYT	56,195	5,000	5,000	3,266	1,297	978	24
WebNLG	5,019	500	703	216	457	26	216

Table 2 Statistics of the two datasets. The number of samples in the test set that belongs to each category, is also reported. Note that a sample can belong to both the *SEO* and *EPO* categories. In addition, the relation number of the WebNLG was miswritten as 246, as in (Fu et al. 2019; Yu et al. 2020), which is the total number of relations in the original WebNLG dataset instead of the number of the subsets they used. We recounted and provided the correct number.

Specifically, a predicted triplet is correct if and only if its relation type and its two corresponding entities are all the same as those in the gold standard annotation. The results of the test set were reported when the development set achieved the best result.

3.1.2 Implementation Details

We implemented the neural networks using the PyTorch library.² Batch padding was applied to pad the lengths of all tokens to make them equal to the maximum length in each batch. The mini-batch training size was set to 64, which was selected from the set: [32, 50, 64].

We used the 300-dimensional GloVe (Pennington et al. 2014) to initialize the word embeddings. Each word representation was concatenated by three parts: pre-trained GloVe embedding, character-based word representation by running a CNN on the character sequence of the word, and POS embedding. The POS, character, and position embeddings were randomly initialized with 30 dimensions (selected from the set: [30, 40, 50]). The filter size of the CNN was set to 3 from the set: [3, 4, 5], and the number of filters was 50 from the set: [30, 40, 50]. Thus, the representation of each word had a dimensionality of 380 (as the input of the BiLSTM layer). For the BiLSTM layer, the hidden vector size was set to 200 from the set: [150, 200]. The *Adam* optimizer (Kingma and Ba 2014) with a learning rate 0.0001 from the set: [0.0001, 0.00001] was employed for training. Dropout was applied to word embeddings and hidden states at a rate of 0.4 from the set: [0.3, 0.4]. We also set the gradient clip-norm to 5 to prevent the gradient explosion problem. The threshold γ was set to 11 for the NYT training set and to 7 for the WebNLG training set. In addition, the value of α in the final loss function (Eq. 15) was set to 0.3 on the NYT and to 0.2 on the WebNLG, where α was in the range (0, 1]. We trained the model for 100 epochs on both datasets. Hyperparameters were tuned on the development set. All experiments were run on a Tesla V100 graphics card in an Ubuntu-based computer system.

3.2 Experimental Results and Analyses

3.2.1 Comparison Models

For comparison, we employed the following models as baselines:

- **NovelTagging** (Zheng et al. 2017): The first model to introduce a novel tagging scheme that transforms the joint extraction task into a sequence labeling problem.
- **MultiDecoder** (Zeng et al. 2018): A seq2seq model with a copy mechanism that converts the joint extraction task to a sequence-to-sequence problem.

² PyTorch is an open-source software library for machine intelligence: <https://pytorch.org/>

- **MultiHead** (Bekoulis et al. 2018): A joint neural model that performs entity recognition and relation extraction simultaneously.
- **GraphRel** (Fu et al. 2019): An end-to-end relation extraction model that uses GCNs to jointly learn named entities and relations.
- **OrderRL** (Zeng et al. 2019): A sequence-to-sequence model with reinforcement learning that takes the extraction order into consideration.
- **ETL-Span** (Yu et al. 2020): A sequence labeling framework based on a novel decomposition strategy that has achieved a notable performance; however, its decomposition strategy still cannot solve the EPO problem, as the authors stated.

3.2.2 Main Results

Table 3 shows the results of our models against those of other baseline methods on both the NYT and WebNLG datasets. First, ETL-Span (Yu et al. 2020) with a decomposition strategy significantly outperformed the previous works by a wide margin. However, because this approach cannot solve the EPO problem as Yu et al. (2020) stated, further improvement of the system performance is hindered. Meanwhile, our model framework with a new decomposition strategy overcomes the existing problems of the model of Yu et al. (2020) and substantially boosts the system performance. Specifically, our approach improved the F1-score by 7.1 on the NYT and by 2.9 on the WebNLG, compared with the results of Yu et al. (2020).

3.2.3 Analysis of Our Decomposition Strategy

To gain more insight into the improvement of our decomposition strategy in our model (in Figure 3), we conducted further experiments, as reported in Table 4. We also reproduced the

Model	NYT			WebNLG		
	<i>Precision</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Recall</i>	<i>F1</i>
NovelTagging (Zheng et al. 2017)	32.8	30.6	31.7	52.5	19.3	28.3
MultiDecoder (Zeng et al. 2018)	61.0	56.6	58.7	37.7	36.4	37.1
MultiHead (Bekoulis et al. 2018)	60.7	58.6	59.6	57.5	54.1	55.7
GraphRel (Fu et al. 2019)	63.9	60.0	61.9	44.7	41.1	42.9
OrderRL (Zeng et al. 2019)	77.9	67.2	72.1	63.3	59.9	61.6
ETL-Span (Yu et al. 2020)	85.5	71.7	78.0	84.3	82.0	83.1
Ours	82.2	88.2	85.1	84.3	87.7	86.0

Table 3 Main results of the performances of the compared models on the NYT and WebNLG.

Model	NYT			WebNLG		
	<i>Precision</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Recall</i>	<i>F1</i>
ETL-Span*	84.4	72.2	77.8	84.5	81.6	83.0
(a) Ours (multiclass)	81.4	76.6	78.9	83.3	86.9	85.1
(b) Ours (multilabel)	81.3	83.7	82.4	81.4	85.3	83.3
(c) Ours (multiclass + <i>URLs</i>)	82.2	88.2	85.1	84.3	87.7	86.0

Table 4 Analysis of the performance of our framework on the test sets. The * marks the results that we reproduced. The *URLs* are the set of “*unified relation labels*” created using Algorithm 1.

results of ETL-Span (Yu et al. 2020).

First, for case (a) in Table 4, we considered our model without using the set *URLs*. Specifically, we used multiclass classifiers on the set of original relations in the two submodules of the module *HTER Extractor*. Compared with the model of Yu et al. (2020), our model achieved gains of 1.1 and 2.1 in the *F1*-score on the NYT and WebNLG, respectively, using only the set of original relations. The model of (Yu et al. 2020) is too strict in regard to the order it obtains the elements of each triplet, as it always detects the head entities first and then extracts the corresponding tail entities and relations for the given HE. Consequently, it will miss all triplets related to an omitted valid HE. Meanwhile, it is not always easy to extract head entities first for all relations, as in some cases it might be easier to detect the tail entities first before the head entities. Thus, our flexible approach overcomes this problem and significantly improves the recall. Note that our approach achieved a better improvement in the *F1*-score on the WebNLG than that on the NYT. One possible reason is that, because the number of relation types in the WebNLG (216 types) is much larger than that in the NYT (only 24 types), it increased the probability of relations where it was easier to detect the tail entities first before the head entities.

Second, as our multiclass model in case (a) cannot solve the EPO problem, we considered the first solution. Specifically, in case (b), we used multilabel classifiers, instead of multiclass classifiers, on the set of original relations in the two submodules of the *HTER Extractor*. With this solution, each tagging position in the *HTER Extractor* can hold multiple original relation types. Thus, we can extract multiple relations (if any) of the same entity pair. By doing this, compared to case (a), our system achieved a gain of 3.5 in the *F1*-score on the NYT, whereas it showed a decreased of 1.8 in the *F1*-score on the WebNLG. We observed that the main difference between the NYT and WebNLG might have led to this result. Specifically, the number of original relations in the WebNLG (216 types) is much larger than that in the NYT (24 types), although

the maximum number of relations of the same entity pair is 3 on both of these training sets. Consequently, the sparse label problem of the multilabel classification on the same entity pair is more severe in the WebNLG than in the NYT. Therefore, it considerably affected the system performance on the WebNLG. Meanwhile, although this problem is less severe in the NYT than in the WebNLG, it also hinders the further improvement of the system performance.

Finally, as our model suffers from the sparse label problem for multilabel classification of the same entity pair in case (b), we considered the second solution to solve the EPO problem. Specifically, in case (c), because a multiclass classification can alleviate the sparse label problem, we used multiclass classifiers with the *URLs* created using Algorithm 1 in the *HTER Extractor*. Interestingly, by using this simple solution, we achieved the highest system performance for both the NYT and WebNLG. Compared with case (a), the solution increased the *F1*-score by 6.2 and 0.9 on the NYT and WebNLG, respectively. It is worth mentioning that the improvement gain on the NYT was significantly larger than that on the WebNLG. One possible reason is that the EPO problem on the NYT is more serious than that on the WebNLG. In Table 2, the number of samples belonging to the EPO category in the NYT test set is 978 (19.6%), whereas it is only 26 (3.7%) in the WebNLG test set.

Compared with the ETL-Span model by Yu et al. (2020), in Table 4, our best model (case (c)) achieved a significant improvement of the system performance with an increase in the *F1*-score by 7.3 and 3.0 on the NYT and WebNLG test sets, respectively. In addition, on the NYT test set, compared with the ETL-Span model, although our best model boosted the recall significantly by 16%, the precision decreased by 2%. One possible reason for the decrease in the precision is that our model tries to train all three parts (i.e., *Entity Extractor* and the two submodules: *TER* and *HER*) effectively at the same time, which might be more challenging than training only two elements simultaneously (i.e., the *HE Extractor* and *TER Extractor*), as in the ETL-Span model of Yu et al. (2020). In future work, we plan to design model architectures more effectively, to obtain a satisfactory level of not only the recall measure but also the precision measure, thereby further improving the *F1*-score.

3.2.4 Analysis of Different Sample Types

To verify the capability of our model to extract multiple triplets, we followed the procedure in (Zeng et al. 2018; Fu et al. 2019) and conducted further experiments on the NYT test set. Specifically, we first split the samples in this test set into three categories: *Normal*, *EPO*, and *SEO*, and then we investigated the performance of each category.

The results are shown in Figure 4. It can be seen from the figure that the performance

Fig. 4 F1-score of extracting relational triplets from samples in three different categories on the NYT test set.

improvement in our model mainly comes from its ability to deal with the EPO and SEO problems more effectively. Compared with the model of Yu et al. (2020), our model achieved competitive performance in all the three categories. In addition, we paid special attention to the performance differences between our approach and that of Yu et al. (2020). Notably, on the NYT test set, our approach boosted the *F1*-score significantly in the EPO problem by 27.0, whereas that of (Yu et al. 2020) cannot solve this problem. In addition, as the strategy of Yu et al. (2020) strictly constrains the detection of the entities to the head first, if it fails to find a valid HE, it will then miss the related triplets. It will be more serious if this HE attends many different triplets in the *head* entity role (a case of SEO). Therefore, our flexible approach deals with this issue and substantially improves the *F1*-score by 3.4 in the SEO problem.

We also compared the ability of the models to extract multiple triplets in a sample. Specifically, we divided the samples of the NYT test set into five categories, where each category contains samples that have 1, 2, 3, 4, or ≥ 5 triplets, respectively. The results are shown in Figure 5. It can be seen from the figure that our approach achieved a significant improvement in extracting multiple triplets compared with the other models. In particular, our model showed a more stable performance when the number of triplets in the sample increased. These results show that our approach is effective in dealing with the multi-relation extraction task.

Fig. 5 F1-scores obtained after extracting relational triplets from samples with different numbers of triplets on the NYT test set.

3.3 Case Study

To gain more insight into the effectiveness of our model in overcoming the existing disadvantages in the approach of Yu et al. (2020), we analyzed the prediction outputs of both models on a few samples of the NYT and WebNLG test sets and these are shown in Tables 5 and 6, respectively.

3.3.1 Dealing With the EPO Problem

In Table 5, we show two examples from the NYT test set and compare the predicted triplets of the model of Yu et al. (2020) with those of our model.

As mentioned earlier, the model of Yu et al. (2020) cannot solve the EPO problem. For any entity pairs, their model only predicts a single relation, although an entity pair can have multiple relations. For instance, in Sample 1, the ordered entity pair (“*Somalia*”, “*Mogadishu*”) has two relations: “/location/country/capital” and “/location/location/contains”. However, the model of Yu et al. (2020) extracted only a single relation and created the triplet: (“*Somalia*”, “/location/location/contains”, “*Mogadishu*”). Similarly, in Sample 2, although the ordered entity pair (“*Ethiopia*”, “*Addis Ababa*”) has three relations: “/location/country/capital”, “/location/country/administrative_divisions”, and “/location/location/contains”, their model predicted only the relation “/location/location/contains” for this pair. Thus, the more serious the EPO problem is, the more degraded the system performance becomes. Meanwhile, our model overcomes this disadvantage and effectively solves the EPO problem. For both samples above, our model fully detected all possible relations for the pair (“*Somalia*”, “*Mogadishu*”) in Sample

Sample 1	Anti-Ethiopia riots erupted in Mogadishu, the capital of Somalia, on Friday, while masked gunmen emerged for the first time on the streets, a day after Ethiopian-backed troops captured the city from Islamist forces.
Yu et al. (2020)	(“ <i>Somalia</i> ”, “/location/location/contains”, “ <i>Mogadishu</i> ”)
Our model	(“ <i>Somalia</i> ”, “/location/country/capital”, “ <i>Mogadishu</i> ”) (“ <i>Somalia</i> ”, “/location/location/contains”, “ <i>Mogadishu</i> ”)
Ground Truth	(“ <i>Somalia</i> ”, “/location/country/capital”, “ <i>Mogadishu</i> ”) (“ <i>Somalia</i> ”, “/location/location/contains”, “ <i>Mogadishu</i> ”)
Sample 2	Though officials in Addis Ababa, Ethiopia’s capital, have said their troops should not enter downtown Mogadishu, many are camped in the former American Embassy, a decrepit building that was closed more than 15 years ago after American soldiers suffered a humiliating defeat at the hands of warlords.
Yu et al. (2020)	(“ <i>Ethiopia</i> ”, “/location/country/capital”, “ <i>Mogadishu</i> ”) (“ <i>Ethiopia</i> ”, “/location/location/contains”, “ <i>Addis Ababa</i> ”) (“ <i>Addis Ababa</i> ”, “/location/administrative.division/country”, “ <i>Ethiopia</i> ”)
Our model	(“ <i>Ethiopia</i> ”, “/location/country/capital”, “ <i>Addis Ababa</i> ”) (“ <i>Ethiopia</i> ”, “/location/country/administrative.divisions”, “ <i>Addis Ababa</i> ”) (“ <i>Ethiopia</i> ”, “/location/location/contains”, “ <i>Addis Ababa</i> ”) (“ <i>Addis Ababa</i> ”, “/location/administrative.division/country”, “ <i>Ethiopia</i> ”)
Ground Truth	(“ <i>Ethiopia</i> ”, “/location/country/capital”, “ <i>Addis Ababa</i> ”) (“ <i>Ethiopia</i> ”, “/location/country/administrative.divisions”, “ <i>Addis Ababa</i> ”) (“ <i>Ethiopia</i> ”, “/location/location/contains”, “ <i>Addis Ababa</i> ”) (“ <i>Addis Ababa</i> ”, “/location/administrative.division/country”, “ <i>Ethiopia</i> ”)

Table 5 Prediction outputs on two samples from the NYT test set.

1 and the pair (“*Ethiopia*”, “*Addis Ababa*”) in Sample 2.

3.3.2 Effect of the “*Exhaustive Search*” Strategy

As shown in Figure 3, our model uses the *Entity Extractor* to detect all entities first. Then, for each detected entity, the *HTER Extractor* utilizes its two submodules to identify the entity in each *head/tail* role and extracts all the corresponding *tail entities/head entities* and relations. The final output of our model is always obtained by combining the results of the two submodules without any duplicate triplets. In essence, this approach can be considered as an “*exhaustive search*” strategy that aims to increase the chances of extracting a valid triplet that may be overlooked by the approach of Yu et al. (2020). Therefore, in Table 6, we compare the prediction outputs of both approaches on three samples from the WebNLG test set.

First, in Sample 3, the *HE Extractor* in the model of Yu et al. (2020) missed the HE “Athens International Airport”, thereby overlooking the valid triplet: (“*Athens International Airport*”,

Sample 3		The Athens International Airport serves the city of Athens , in Greece where Alexis Tsipras is the leader.
Yu et al. (2020)		(“Athens”, “country”, “Greece”) (“Greece”, “leaderName”, “Alexis Tsipras”)
Our model	<i>HER+URLs</i>	(“Athens”, “country”, “Greece”) (“Greece”, “leaderName”, “Alexis Tsipras”)
	<i>TER+URLs</i>	(“Athens”, “country”, “Greece”) (“Greece”, “leaderName”, “Alexis Tsipras”) (“Athens International Airport”, “cityServed”, “Athens”)
Ground Truth		(“Athens”, “country”, “Greece”) (“Greece”, “leaderName”, “Alexis Tsipras”) (“Athens International Airport”, “cityServed”, “Athens”)
Sample 4		Faber and Faber are the publishers of The Secret Scripture, a sequel to A Long Long Way. That book comes from Ireland which is located in Europe and where there is an ethnic group of white people.
Yu et al. (2020)		(“A Long Long Way”, “country”, “Ireland”) (“A Long Long Way”, “followedBy”, “The Secret Scripture”)
Our Model	<i>HER+URLs</i>	(“A Long Long Way”, “country”, “Ireland”) (“A Long Long Way”, “followedBy”, “The Secret Scripture”) (“The Secret Scripture”, “publisher”, “Faber and Faber”)
	<i>TER+URLs</i>	(“A Long Long Way”, “country”, “Ireland”) (“A Long Long Way”, “followedBy”, “The Secret Scripture”)
Ground Truth		(“A Long Long Way”, “country”, “Ireland”) (“A Long Long Way”, “followedBy”, “The Secret Scripture”) (“The Secret Scripture”, “publisher”, “Faber and Faber”) (“Ireland”, “location”, “Europe”)
Sample 5		3Arena is located in Dublin, the Republic of Ireland, where Críona Ní Dhálaigh was Lord Mayor. The owner of 3Arena is Live Nation Entertainment.
Yu et al. (2020)		(“Dublin”, “country”, “Republic of Ireland”) (“Dublin”, “leaderName”, “Críona Ní Dhálaigh”) (“Dublin”, “leaderName”, “Lord Mayor”)
Our model	<i>HER+URLs</i>	(“3Arena”, “location”, “Dublin”) (“3Arena”, “owner”, “Live Nation Entertainment”) (“Dublin”, “country”, “Republic of Ireland”)
	<i>TER+URLs</i>	(“Dublin”, “country”, “Republic of Ireland”) (“Dublin”, “leaderName”, “Críona Ní Dhálaigh”)
Ground Truth		(“3Arena”, “location”, “Dublin”) (“3Arena”, “owner”, “Live Nation Entertainment”) (“Dublin”, “country”, “Republic of Ireland”) (“Dublin”, “leaderName”, “Críona Ní Dhálaigh”)

Table 6 Prediction outputs on a few samples from the WebNLG test set.

“cityServed”, “Athens”) in the ground truth. Meanwhile, in our model, the entity “Athens International Airport” was detected by the *Entity Extractor*. Then, the *TER+URLs* submodule of the *HTER Extractor* identified this entity in the *head* role and extracted the triplet (“Athens International Airport”, “cityServed”, “Athens”). Additionally, we compared the outputs of the *HER+URLs* and *TER+URLs* submodules in our model. Although two triplets, namely, (“Athens”, “country”, “Greece”) and (“Greece”, “leaderName”, “Alexis Tsipras”), were easily obtained by the two submodules, the *HER+URLs* submodule failed to extract the valid triplet: (“Athens International Airport”, “cityServed”, “Athens”) when considering the entity “Athens” in the *tail* role. Thus, in this example, the *TER+URLs* submodule achieved a better result than that of the *HER+URLs* submodule.

Second, in Sample 4, the approach of Yu et al. (2020) omitted two valid triplets in the ground truth: (“The Secret Scripture”, “publisher”, “Faber and Faber”) and (“Ireland”, “location”, “Europe”), because the *HE Extractor* missed two HEs: “The Secret Scripture” and “Ireland”. In our model, although the module *Entity Extractor* could detect the entity “The Secret Scripture”, its *TER+URLs* submodule failed to extract the triplet (“The Secret Scripture”, “publisher”, “Faber and Faber”) when considering “The Secret Scripture” in the *head* role. Meanwhile, thanks to the *HER+URLs* submodule, it extracted this missed triplet by considering “Faber and Faber” in the *tail* role and detecting the corresponding HE “The Secret Scripture” with the relation type “publisher”. Based on the outputs of the two submodules, it is clear that the *HER+URLs* submodule yielded a better result for this sample than that of the *TER+URLs* submodule.

Finally, in Sample 5, the model of Yu et al. (2020) obtained only two valid triplets in the ground truth: (“Dublin”, “country”, “Republic of Ireland”) and (“Dublin”, “leaderName”, “Críona Ní Dhálaigh”). The *HE Extractor* of their model missed the HE “3Arena”, thereby overlooking the triplets: (“3Arena”, “location”, “Dublin”) and (“3Arena”, “owner”, “Live Nation Entertainment”). In our model, the *Entity Extractor* also missed the entity: “3Arena”. Consequently, the *TER+URLs* submodule also overlooked the triplets: (“3Arena”, “location”, “Dublin”) and (“3Arena”, “owner”, “Live Nation Entertainment”). Meanwhile, the *HER+URLs* submodule identified the entity “Dublin” in the *tail* role to extract the triplet (“3Arena”, “location”, “Dublin”) and also identified the entity “Live Nation Entertainment” in the *tail* role to extract the triplet (“3Arena”, “owner”, “Live Nation Entertainment”). However, the *HER+URLs* submodule missed the valid triplet (“Dublin”, “leaderName”, “Críona Ní Dhálaigh”), whereas this triplet was detected by the *TER+URLs* submodule. Our model obtained the final result by combining the outputs of the two submodules.

We further consider the system performance of the predicted outputs of the (*HER+URLs*

and *TER+URLs*) submodules of the *HTER Extractor* of our model on the entire WebNLG test set in Table 7. We can see that the number of predicted triplets by the *HER+URLs* submodule is 1,510, whereas this number is 1,530 for the *TER+URLs* submodule. In addition, these two submodules share 1,368 common predicted triplets. Thus, the overlap percentage of the output of the *HER+URLs* submodule is 90.6, whereas this rate is 89.4 for the output of the *TER+URLs* submodule. In Table 7, our model achieved the best performance when combining the predicted outputs of the two submodules.

On the basis of the results of the analysis of the examples in Table 6 and of the performances of the submodules of our model in Table 7, we conclude that the “*exhaustive search*” strategy of our model is effective in solving the entity and relation extraction task.

3.4 Impact of Using Pre-trained Language Models

For a fair comparison, like in Yu et al. (2020), we did not exploit the advantages of using pretrained language models. In reality, a well-known pretrained language model named BERT was first proposed by Devlin et al. (2019). It has been widely applied to various NLP downstream tasks and has achieved considerable success. For the entity and relation extraction task, Hang et al. (2021) presented a BERT-based model named BERT-JEORE and obtained superior performance. Therefore, we further investigated the impact of using pretrained language models when they were used in our model.

Specifically, for our model in Figure 3, we replaced only the first BiLSTM encoder with a pretrained BERT-Base encoder to extract the representations of the original words from the input sample. Note that the BERT model first uses its tokenizer to split each original word into tokens (if necessary) and then outputs the vectors of these tokens. Thus, we obtained the representation of each original word by averaging its start token vector and its end token vector. In Table 8, we report the system performance of our model on the NYT and WebNLG datasets.

Clearly, our model showed further performance boost on both the NYT and WebNLG datasets

<i>F1</i>	
<i>HER+URLs</i>	85.1
<i>TER+URLs</i>	85.3
<i>Combined</i>	86.0

Table 7 Results of the performance analysis of the two submodules of our model on the WebNLG test set. The set *URLs* was created using Algorithm 1.

Model	NYT			WebNLG		
	<i>Precision</i>	<i>Recall</i>	<i>F1</i>	<i>Precision</i>	<i>Recall</i>	<i>F1</i>
BERT-JEORE (Hang et al. 2021)	88.5	84.6	86.5	79.1	91.4	84.8
Ours_{LSTM}	82.2	88.2	85.1	84.3	87.7	86.0
Ours_{BERT}	90.7	92.6	91.6	85.0	87.9	86.4

Table 8 Impact of using a pretrained BERT encoder in our model.

when employing a pretrained BERT encoder, significantly improving the $F1$ -score by 6.5 on the NYT dataset. Compared to a recent model based on BERT (BERT-JEORE) by Hang et al. (2021) for the entity and relation extraction task, our BERT-based model achieved a better performance on the NYT and WebNLG datasets, with gains of 5.1 and 1.6 in the $F1$ -score, respectively. In addition, it is interesting to note that, even without using the BERT encoder, our model still outperformed the recent model on the WebNLG test set. This indicates that our approach with a new decomposition strategy is simple but very effective in solving the entity and relation extraction task.

4 Related Works

Researchers have made great efforts to extract relational triplets from unstructured text, which can be directly used for automatic knowledge graph construction. Early works (Zelenko et al. 2003; Zhou et al. 2005; Chan and Roth 2011) regarded the joint extraction task in a pipeline manner. They extracted relational triplets in two isolated steps: (1) first by running the NER on the input sample to recognize all entities, and (2) then by running the RC on all pairs of the extracted entities. Although these pipeline methods are quite simple, they usually suffer from the error propagation problem and ignore the relevance between the two steps.

To ease the two issues above, subsequent works attempted to build joint learning models that learn entities and relations simultaneously in a single manner. They can be divided into two main approaches: feature-based models (Yu and Lam 2010; Li and Ji 2014; Miwa and Sasaki 2014; Ren et al. 2017) and neural network-based models (Gupta et al. 2016; Katiyar and Cardie 2017; Zheng et al. 2017; Zeng et al. 2018; Fu et al. 2019; Yu et al. 2020). The former rely heavily on feature engineering and require intensive manual efforts, whereas the latter are mainly based on neural network architectures. Zheng et al. (2017) introduced a unified tagging scheme and converted the joint entity and relation extraction task to an end-to-end sequence tagging problem. This method

can directly model relational triplets as a whole at the triplet level because the unified tagging scheme already integrates the information of both entities and relations. However, most previous studies ignored the problem of overlapping relational triplets. Zeng et al. (2018) presented three patterns of overlapping triplets and made an effort to address the problem via a sequence-to-sequence model with a copy mechanism. Subsequently, Fu et al. (2019) also focused on this problem and proposed a GCN-based method to address it. Recently, Yu et al. (2020) introduced a novel decomposition strategy that decomposes the task into *HE* and *TER* extractions, where the *HE* extractor detects head entities and the *TER* extractor identifies the corresponding tail entity and relation for each given HE. Although this approach significantly outperforms previous works, it still cannot solve the EPO problem as Yu et al. (2020) stated in their work.

On a related note, pretrained language models have also been exploited for entity and relation extraction, thereby utilizing prior knowledge and achieving superior results. Zhao et al. (2020) adopted BERT (Devlin et al. 2019) as the machine reading comprehension model to solve the joint entity and relation extraction. Hang et al. (2021) proposed BERT-JEORE, an end-to-end neural network model that is based on BERT for the joint extraction of entities and overlapping relations. They used source-target BERT to generate an entity label for each token in the sample and utilized an overlapping relation extraction model to create an unlimited number of relational triplets.

5 Conclusion and Future Work

This paper proposes a new decomposition strategy along with a corresponding model framework for the joint entity and relation extraction task. Our approach mainly focuses on solving the overlapping triplet problem, one of the biggest challenges of this task, as only a few existing works can tackle this problem effectively. Our model uses a module to extract all the relevant entities, and for each extracted entity, another module is utilized to consider its *head/tail* entity roles and extract all the related triplets. In addition, the use of *URLs* helps to sufficiently deal with the sparse label problem of relation types in the same entity pair (e.g., EPO cases), which can be prevalent in this task. Experimental results on the two widely used datasets (NYT and WebNLG) showed that our model achieved a notable performance compared with a recent work (Hang et al. 2021). The results of further analysis experiments showed the effectiveness of our approach in handling overlapping and multiple triplet extraction scenarios.

Our proposed methodology has considerable potential for practical NLP applications such as information extraction, knowledge base population, and question answering. Moreover, the idea

of using *URLs* may be relevant and promising for multilabel classification problems in general, not just for a specific task such as the entity and relation extraction task. In future work, we plan to apply this idea to the text classification task. Additionally, we also would like to introduce other methods for solving the overlapping triplet problem more effectively, such as considering how to change the weight of a label depending on whether it is a subset of the true label, and integrating available knowledge bases of entities into current models for boosting the system performance.

References

- Bekoulis, G., Deleu, J., Demeester, T., and Develder, C. (2018). “Joint Entity Recognition and Relation Extraction as a Multi-head Selection Problem.” *Expert Systems with Applications*, 114, pp. 34–45.
- 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*, pp. 551–560, Portland, Oregon, USA. 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*, Vol. 33, pp. 6300–6308.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- 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 Annual Meeting of the Association for Computational Linguistics*, pp. 1409–1418, Florence, Italy. Association for Computational Linguistics.
- Gardent, C., Shimorina, A., Narayan, S., and Perez-Beltrachini, L. (2017). “Creating Training Corpora for NLG Micro-Planners.” In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 179–188, Vancouver, Canada. 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*, pp. 2537–2547, Osaka, Japan. The COLING 2016 Organizing Committee.
- Hang, T., Feng, J., Wu, Y., Yan, L., and Wang, Y. (2021). “Joint Extraction of Entities and Overlapping Relations using Source-target Entity Labeling.” *Expert Systems with Applications*, **177**. 114853.
- Hao, Y., Zhang, Y., Liu, K., He, S., Liu, Z., Wu, H., and Zhao, J. (2017). “An End-to-End Model for Question Answering over Knowledge Base with Cross-Attention Combining Global Knowledge.” In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 221–231, Vancouver, Canada. Association for Computational Linguistics.
- Hochreiter, S. and Schmidhuber, J. (1997). “Long Short-Term Memory.” *Neural Computation*, **9** (8), pp. 1735–1780.
- Katihar, A. and Cardie, C. (2017). “Going out on a limb: Joint Extraction of Entity Mentions and Relations without Dependency Trees.” In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 917–928, Vancouver, Canada. Association for Computational Linguistics.
- 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)*, pp. 402–412, Baltimore, Maryland. Association for Computational Linguistics.
- Miwa, M. and Bansal, M. (2016). “End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures.” In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1105–1116, Berlin, Germany. Association for Computational Linguistics.
- 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)*, pp. 1858–1869.
- Ouchi, H., Shindo, H., and Matsumoto, Y. (2018). “A Span Selection Model for Semantic Role Labeling.” In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1630–1642, Brussels, Belgium. Association for Computational Linguistics.
- Pennington, J., Socher, R., and Manning, C. (2014). “GloVe: Global Vectors for Word Represen-

- tation.” In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Ren, X., Wu, Z., He, W., Qu, M., Voss, C. R., Ji, H., Abdelzaher, T. F., and Han, J. (2017). “Co-Type: Joint Extraction of Typed Entities and Relations with Knowledge Bases.” In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*, pp. 1015–1024, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.
- Riedel, S., Yao, L., and McCallum, A. (2010). “Modeling Relations and Their Mentions without Labeled Text.” In Balcázar, J. L., Bonchi, F., Gionis, A., and Sebag, M. (Eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 148–163, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Tran, V.-H., Phi, V.-T., Shindo, H., and Matsumoto, Y. (2019). “Relation Classification Using Segment-Level Attention-based CNN and Dependency-based RNN.” In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2793–2798, Minneapolis, Minnesota. Association for Computational Linguistics.
- Tran, V., Tran, V.-H., Nguyen, P., Nguyen, C., Satoh, K., Matsumoto, Y., and Nguyen, M. (2021). “CovRelex: A COVID-19 Retrieval System with Relation Extraction.” In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pp. 24–31, Online. Association for Computational Linguistics.
- Vu, N.-T., Tran, V.-H., Doan, T.-H.-T., Le, H.-Q., and Tran, M.-V. (2015). “A Method for Building a Labeled Named Entity Recognition Corpus Using Ontologies.” In Le Thi, H. A., Nguyen, N. T., and Do, T. V. (Eds.), *Advanced Computational Methods for Knowledge Engineering*, pp. 141–149, Cham. Springer International Publishing.
- Yu, B., Zhang, Z., Shu, X., Liu, T., Wang, Y., Wang, B., and Li, S. (2020). “Joint Extraction of Entities and Relations Based on a Novel Decomposition Strategy.” *Frontiers in Artificial Intelligence and Applications*, **325** (ECAI 2020), pp. 2282–2289.
- Yu, X. and Lam, W. (2010). “Jointly Identifying Entities and Extracting Relations in Encyclopedia Text via A Graphical Model Approach.” In *Coling 2010: Posters*, pp. 1399–1407, Beijing, China. Coling 2010 Organizing Committee.
- Zelenko, D., Aone, C., and Richardella, A. (2003). “Kernel Methods for Relation Extraction.” *Journal of Machine Learning Research*, **3**, pp. 1083–1106.

- Zeng, X., He, S., Zeng, D., Liu, K., Liu, S., and Zhao, J. (2019). “Learning the Extraction Order of Multiple Relational Facts in a Sentence with Reinforcement Learning.” In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 367–377, Hong Kong, China. Association for Computational Linguistics.
- 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)*, pp. 506–514, Melbourne, Australia. Association for Computational Linguistics.
- Zhao, T., Yan, Z., Cao, Y., and Li, Z. (2020). “Asking Effective and Diverse Questions: A Machine Reading Comprehension Based Framework for Joint Entity-relation Extraction.” In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 3948–3954.
- 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)*, pp. 1227–1236, Vancouver, Canada. Association for Computational Linguistics.
- Zhou, G., Su, J., Zhang, J., and Zhang, M. (2005). “Exploring Various Knowledge in Relation Extraction.” In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*, pp. 427–434, Ann Arbor, Michigan. Association for Computational Linguistics.

Van-Hien Tran: Van-Hien Tran received the B.E. degree in information technology from the University of Engineering and Technology, Vietnam National University (VNU), Vietnam, in 2014, and the M.E. degree from the Nara Institute of Science and Technology (NAIST), Japan, in 2019, where he is currently pursuing the Ph.D. degree. His research interests include machine learning and natural language processing, in particular on information extraction.

Van-Thuy Phi: Van-Thuy Phi is currently a postdoctoral researcher at the Nara Institute of Science and Technology (NAIST). Previously, he received the M.Sc. degree in 2016, and the Ph.D. degree in 2019, both in Information Science from NAIST. His research interests lie at the intersection of Natural Language Processing and Computer Vision, in particular on areas relating to

information extraction from texts or images, image segmentation and deep learning.

Akihiko Kato: Akihiko Kato received the Ph.D. degree in Engineering from Nara Institute of Science and Technology (NAIST), in 2019. He is currently a postdoctoral researcher in Graduate School of Information Science, NAIST. His research interests include information extraction, knowledge base completion, and syntactic/semantic parsing.

Hiroyuki Shindo: Hiroyuki Shindo received the B.E. and M.E. degrees from Waseda University, Japan, in 2007 and 2009, respectively, and the Ph.D. degree in engineering from the Nara Institute of Science and Technology (NAIST), Ikoma, Japan, in 2013. From 2009 to 2014, he was a Researcher with NTT Communication Science Laboratories. He is currently an Assistant Professor with the Graduate School of Information Science, NAIST. His research interests include machine learning and computational linguistics.

Taro Watanabe: Taro Watanabe received the B.E. and M.E. degrees in information science from Kyoto University, Kyoto, Japan in 1994 and 1997, respectively, and obtained the Master of Science degree in language and information technologies from the School of Computer Science, Carnegie Mellon University in 2000. In 2004, he received the Ph.D. in informatics from Kyoto University, Kyoto, Japan. After working as a researcher at ATR, NTT and NICT, and as a software engineer at Google, he is a professor at Nara Institute of Science and Technology starting from 2020. His research interests include natural language processing, machine learning and machine translation.

Yuji Matsumoto: Yuji Matsumoto is currently a team leader at RIKEN Center for Advanced Intelligence Project (AIP). He received his M.S. and Ph.D. degrees in information science from Kyoto University in 1979 and in 1990, respectively. He joined Machine Inference Section of Electrotechnical Laboratory in 1979. He has been an academic visitor at Imperial College of Science and Technology, the deputy chief of First Laboratory at ICOT, an associate professor at Kyoto University, and a professor at Nara Institute of Science and Technology. His main research interests are natural language understanding and machine learning. He is an ACL fellow and a fellow of Information Processing Society of Japan.

(Received March 1, 2021)

(Revised June 1, 2021)

(Accepted July 6, 2021)