24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

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

Bowen Yu[1,2] and **Zhenyu Zhang**[1,2] and **Xiaobo Shu**[1,2] and **Tingwen Liu**[1*]
Yubin Wang[1,2] and **Bin Wang**[3] and **Sujian Li**[4]

Abstract. Joint extraction of entities and relations aims to detect entity pairs along with their relations using a single model. Prior work typically solves 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 interrelated 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 5.2%, 5.9% and 21.5% (F1 score), achieving a new state-of-the-art on three public datasets.

1 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) [31]. 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 [13]. One improved way is to train them jointly by parameter sharing [4, 17, 23]. 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* [3, 26], 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. [35] proposed a unified tagging scheme to transform joint extraction to a sequence labeling problem with a kind of multi-part tags. However, this model lacks the elegance to identify overlapping relations, which may lead to poor recall when processing a sentence with overlapping relations. As the improvement, Dai et al. [3] presented PA-LSTM which tags entity and relation labels simultaneously according to each query word position, and achieves state-of-the-art performance. Nevertheless, *these models 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 cannot fully perceive the semantics of head-entity, it will be unreliable to extract the corresponding tail entities and relations.

For a complex NLP task, it is very common to decompose the task into easier modules or processes, and a reasonable design is quite crucial to help one model make further progress [8, 15, 33]. 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, which hierarchically decomposes the task 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. Besides, compared with the extract-then-classify methods, *our paradigm no longer extracts all entities at the first step, only head-entities that are likely to participate in target triplets are identified, thus alleviating the impact of redundant entity pairs*.

Next, inspired by extractive question answering which identifies answer span by predicting its start and end indices [22], 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 posi-

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. Emails: {yubowen, zhangzhenyu1996, shuxiaobo, liutingwen, wangyubin}@iie.ac.cn. * Corresponding Author.
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China.
[3] Xiaomi AI Lab, Xiaomi Inc., China. Email: wangbin11@xiaomi.com.
[4] Key Laboratory of Computational Linguistics, Peking University, MOE, China. Email: lisujian@pku.edu.cn.

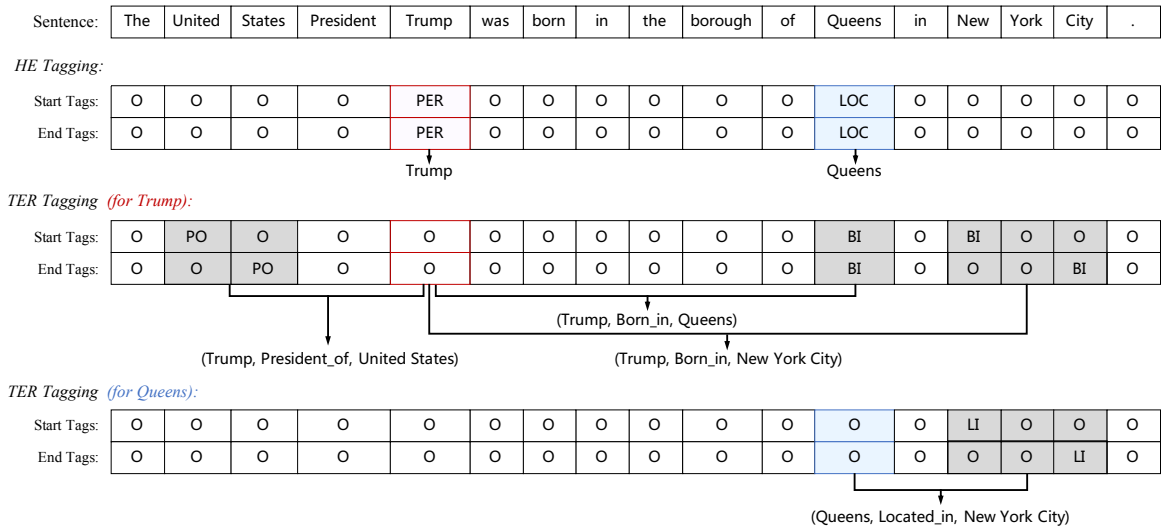

Figure 1. An example of our tagging scheme. *PER* is short for entity type *PERSON*, *LOC* is short for *LOCATION*, *PO* is short for relation type *President_of*, *BI* is short for *Born_in*, and *LI* is short for *Located_in*.

tions 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 compared with the whole extraction task, suggesting that by trained cooperatively with shared underlying representations, they can constrain the learning problem and achieve a better overall outcome.

We evaluate our method on three public datasets: NYT-single, NYT-multi and WebNLG. Experimental results show that the proposed method significantly outperforms previous work on normal, overlapping and multiple relation extraction, increasing the SOTA F1 score to 59.0% (+5.2%), 78.0% (+5.9%) and 83.1% (+21.5%), respectively. Further analysis confirms the effectiveness and rationality of our decomposition strategy.

2 METHODOLOGY

In this section, we first introduce our tagging scheme, based on which the joint extraction task is transformed into 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.

2.1 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 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 that the entity type is labeled for the token which is the end word.

For each identified head-entity, TER extraction 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.

Figure 1 illustrates an example of our tagging scheme, in which the words "*United*", "*States*", "*Trump*", "*Queens*", "*New*" and "*City*" are all related to final extraction results, thus they are labelled with special tags. For example, the word "*Trump*" is the first and also the last word of head-entity "*Trump*", so the tags are both *PERSON* in the start and end tag sequences when tagging HE. For TER extraction, when the given head-entity is "*Trump*", there are two tail-entities involved in with wanted relations, 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 sequences. Similarly, we can obtain end tag sequences 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 [3]. 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.

2.2 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 do not 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 universally modeled as:

$$p(t, l|S) = p(s_t^l|S)p(e_t^l|s_t^l, S) \quad (1)$$

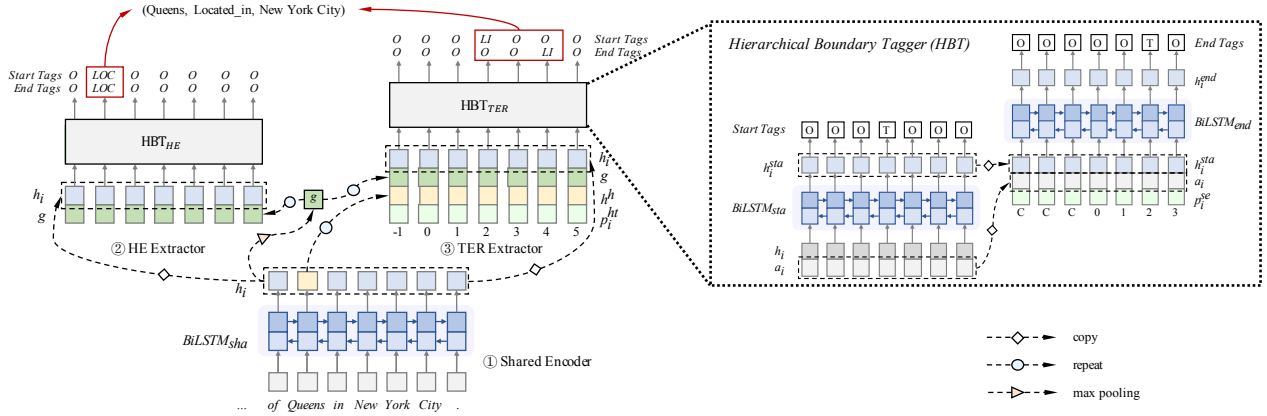

Figure 2. An illustration of our model. The left panel is an overview of our joint extraction system, and the right panel shows the detailed structure of our sequence tagger HBT. Here, "*Queens*" is extracted by the HE extractor, then its hidden state in the shared encoder is marked as the yellow box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

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 [7] as the basic encoder. Formally, the label of word x_i when tagging the start position is predicted as Eq. 4.

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

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

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

where $\mathbf{h}_i$ denotes token representation and $\mathbf{a}_i$ is an auxiliary vector. For HE extraction, $\mathbf{a}_i$ is a global representation learned from the entire sentence. It is beneficial to make more accurate predictions from a global perspective. For TER extraction, $\mathbf{a}_i$ is the concatenation of a global representation and 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.

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

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

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

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 [29], 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

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

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 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 $\text{sta\_tag}(S)$ and $\text{end\_tag}(S)$. We first traverse $\text{sta\_tag}(S)$ 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 $\text{end\_tag}(S)$ from this index to search for a end 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 $\text{end\_tag}(S)$ is terminated to continue to traverse $\text{sta\_tag}(S)$ to find the next start position (Line 20). Once all the indices in $\text{sta\_tag}(S)$ are iterated, this decoding function ends by returning the recordset $\mathbf{R}$ (Line 21).

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 the type tag

Define $n \leftarrow$ Sentence Length
Initialize $\mathbf{R} \leftarrow \{\}$
Initialize $s^* \leftarrow 0$
Initialize p^{se} as a list of length n with default value C
Obtain sta_tag(S) by Eq. 4
for $idx \leftarrow 1$ to n **do**
  if sta_tag $(S)[idx] \neq$ "*O*" **then**
    $s^* \leftarrow idx$
  if $s^* > 0$ **then**
    $p^{se}[idx] \leftarrow idx - s^*$
Obtain $\mathbf{p}^{se}$ by transforming p^{se} into matrix
Obtain end_tag(S) by Eq. 7
for $idx_s \leftarrow 1$ to n **do**
  if sta_tag$(S)[idx_s] \neq$ "*O*" **then**
    for $idx_e \leftarrow idx_s$ to n **do**
      if end_tag$(S)[idx_e] =$ sta_tag$(S)[idx_s]$ **then**
        $e \leftarrow S[idx_s : idx_e]$
        $tag \leftarrow$end_tag$(S)[idx_e]$
        $\mathbf{R} \leftarrow \mathbf{R} \cup \{(e, tag)\}$
        Break
return $\mathbf{R}$

2.3 EXTRACTION SYSTEM

With the span-based tagging scheme and the hierarchical boundary tagger, we propose an end-to-end neural architecture (Figure 2) to extract entities and overlapping relations jointly, which first encodes the sentence with a shared BiLSTM encoder. Then, a HE extractor is built 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.

2.3.1 Shared Encoder

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

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

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. Following [4], we also employ part-of-speech (POS) embedding to enrich $\mathbf{x}_i$.

2.3.2 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 representation 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:

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

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

2.3.3 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 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.

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.

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

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 [5]. 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 $\mathbf{R}_{HE}$ to complete the extraction task.

2.3.4 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:

$$\mathcal{L} = \mathcal{L}_{HE} + \mathcal{L}_{TER} \tag{14}$$

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.

3 EXPERIMENTS

3.1 Experimental Settings

3.1.1 Datasets

Following popular choices and previous work [3, 4, 20, 30, 31, 35], We conduct experiments on three benchmark datasets: (1) **NYT-**

[5] Note that $type_h$ is not included in the final output of our extraction system. However, we claim that by predicting entity types, we can implicitly incorporate type information into head-entity representation, which is beneficial to the subsequent TER tagging as our experiment reveals.

Table 1. Statistics of the datasets.

Dataset	NYT-single	NYT-multi	WebNLG
# Relation types	24	24	246
# Training sentences	66,335	56,195	5,019
# Test sentences	395	5,000	703

single is sampled from the New York Times corpus [21] and published by Ren et al [20]. 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. [31] 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) **WebNLG** is proposed by Claire et al. [5] for Natural Language Generation task. We use the dataset pre-processed by Zeng et al [31] and the train set contains 5019 sentences, the test set contains 703 sentences and the validation set contains 500 sentences. Statistics of the datasets are shown in Table 1.

Besides, as suggested in [4, 31], we also divided the test set into three categories: Normal, SingleEntityOverlap (SEO), and EntityPairOverlap (EPO) to verify the effectiveness on extracting overlapping relations. Specifically, 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. A sentence belongs to SEO class if some of its triplets have an overlapped entity and these triplets dont have any overlapped entity pair. Note that a sentence in the EPO set may contain multiple Normal and SEO triplets. We discuss the result for different categories in the detailed analysis.

3.1.2 Evaluation

We follow the evaluation metrics in previous work [3, 4, 20, 30, 31, 35]. A triplet is marked correct if and only if its relation type and two corresponding entities are all correct, where the entity is considered correct if the head and tail offsets are both correct. We adopt the standard micro Precision, Recall and F1 score to evaluate the results.

3.1.3 Implementation Details

We use the 300 dimension Glove [19] to initialize word embeddings. The POS, character and position embeddings are randomly initialized with 30 dimensions. The window size of CNN for character-based word representations is set to 3, and the number of filters is 50. For Bi-LSTM encoder, the hidden vector length is set to 100. Parameter optimization is performed using Adam [11] with learning rate 0.001 and batch size 64. Dropout is applied to word embeddings and hidden states with a rate of 0.4. 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.

3.1.4 Comparison Models

For comparison, we employ the following models as baselines: (1) **Cotype** [20] learns jointly the representations of entity mentions, relation mentions and type labels; (2) **NovelTagging** [35] is the first proposed unified sequence tagger which predicts both entity type and relation class for each word; (3) **MultiDecoder** [31] considers relation extraction as a sequence-to-sequence problem and uses dynamic decoders to extract relation triplets; (4) **MultiHead** [2] first identifies all candidate entities, then perform relation extraction by identifying multiple relations for each entity, these two tasks are trained jointly; (5) **PA-LSTM** [3] is the current best unified labeling method, which tags entity and relation labels simultaneously according to a query word position and achieves the recent state-of-the-art results on the NYT-single dataset; (6) **GraphRel** [4] is the latest extrat-then-classify method, 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; (7) **OrderRL** [30] is the state-of-the-art method on the NYT-multi and WebNLG datasets, which applies the reinforcement learning into a sequence-to-sequence model to generate multiple triplets.

We call our proposed extract-then-label method with span-based scheme as **ETL-Span**. In addition, to access the performance influence of 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 entity positions and types simultaneously, denoted as **ETL-BIES**.

3.2 Experimental Results and Analyses

3.2.1 Main Results

Table 2 reports the results of our models against other baseline methods. It can be seen that our method, ETL-Span, significantly outperforms all other methods and achieves the state-of-the-art F1 score on all three datasets. Over the latest extract-then-classify method GraphRel, ETL-Span achieves substantial improvements of 16.1% and 40.2% in F1 score on the NYT-multi and WebNLG datasets respectively. 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 improves by a relative margin of 5.2% against the strong baseline PA-LSTM. 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; (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. In addition, we find that the results of our model are better than sequence-to-sequence methods like MultiDecoder and OrderRL, it is likely due to the innate restrictions on RNN unrolling, the capacity of generating triplets is limited [4]. Beyond that, we notice that the Precision of our model drops compared with NovelTagging on the NYT-single dataset. One possible reason is that many overlapping relations are not annotated in the manually labeled test data. Following PA-LSTM [3], we add some gold triplets into NYT-single test set and further achieve a large improvement of 12.5% in F1 score and 18.7% in Precision compared with the results in Table 2. Overall, these results indicate that our extraction paradigm which first extracts head-entity then labels corresponding tail-entity and relation can better capture the relational information in the sentence.

We also observe that ETL-Span performs remarkably better than ETL-BIES, we guess it is because ETL-BIES must do additional

Table 2. Main results on three benchmark datasets. Bold marks highest number among all models. ‡ marks results quoted directly from the original papers. † marks results reported in [3] and [31]. * marks results produced with offcial implementation.

Model	NYT-single			NYT-multi			WebNLG		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
CoType‡ [20]	42.3%	51.1%	46.3%	–	–	–	–	–	–
NovelTagging† [35]	**61.5**%	41.4%	49.5%	32.8%	30.6%	31.7%	52.5%	19.3%	28.3%
MultiDecoder‡ [31]	–	–	–	61.0%	56.6%	58.7%	37.7%	36.4%	37.1%
MultiHead* [2]	51.5%	52.8%	52.1%	60.7%	58.6%	59.6%	57.5%	54.1%	55.7%
PA-LSTM‡ [3]	49.4%	59.1%	53.8%	–	–	–	–	–	–
GraphRel‡ [4]	–	–	–	63.9%	60.0%	61.9%	44.7%	41.1%	42.9%
OrderRL‡ [30]	–	–	–	77.9%	67.2%	72.1%	63.3%	59.9%	61.6%
ETL-BIES	51.1%	64.6%	57.2%	84.4%	71.5%	77.4%	83.5%	81.1%	82.3%
ETL-Span	53.8%	**65.1%**	**59.0%**	**85.5%**	**71.7%**	**78.0%**	**84.3** %	**82.0%**	**83.1%**

Table 3. Comparison of test-time speed. Bat/s refers to the number of batches can be processed per second.

Model	NYT-single	NYT-multi	WebNLG
ETL-BIES	10.9 Bat/s	11.2 Bat/s	6.3 Bat/s
ETL-Span	26.1 Bat/s	25.6 Bat/s	23.5 Bat/s

Table 4. An ablation study of ETL-Span on the NYT-multi dev set.

Model	Precision	Recall	F1
ETL-Span	**86.5%**	**73.5%**	**79.5%**
– Char embedding	83.1%	71.2%	76.7%
– Position embedding $\mathbf{p}^{ht}$	81.9%	70.3%	75.7%
– Hierarchical tagging	84.6%	70.7%	77.0%
– Head-entity type tagging	85.8%	72.2%	78.4%
– Joint learning	80.4%	68.9%	74.2%

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 3.7 times. The main reason is that decoding the best chain of labels with CRF requires a significant amount of computing resources especially when the tag space is huge (e.g., on WebNLG with 246 relations and 989 tags). Besides, ETL-Span only takes about 1/4 time per batch and 1/5 GPU memory compared with ETL-BIES during training, which further verdicts the superiority of our span-based scheme.

3.2.2 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, position embedding $\mathbf{p}^{ht}$, hierarchical tagging (by tagging boundary positions at the outmost BiLSTM layer), head-entity type tagging (by tagging 0/1 instead of entity types in the HE extractor) and joint learning (by training HE extractor and TER extractor separately without parameter sharing). From these ablations shown in Table 4, we find that: (1) Consistent with PA-LSTM [3], the character-level representations are helpful to capture the morphological information and deal with OOV words. (2) When we remove $\mathbf{p}^{ht}$, the score drops by 3.8%, 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. (3) Removing the hierarchical tagging structure hurts the result by 2.5% F1 score, which indicates that predicting end positions benefits from the prediction results of start positions. (4) By predicting entity type in the HE extractor, we can implicitly incorporate type information into head-entity representation, which is beneficial to the subsequent TER tagging. (5) Compared with the pipelined manner, joint learning framework brings a remarkable improvement (5.3%) in F1 score, which demonstrates that our HE extractor and TER extractor actually work in the mutual promotion way, and again confirms the effectiveness and rationality of our decomposition strategy.

3.2.3 Analysis on Different Sentence Types

To verify the ability of our model in handling the overlapping problem, following [4, 31], we conduct further experiments on the NYT-multi test set. The results are shown in Figure 3. Among the compared baselines, GraphRel and OrderRL are the latest two models with 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. OrderRL utilizes a sequence-to-sequence model to decode overlapping relations but can decode only the first word of multi-word entity, while ours can detect the whole. Readers may have noticed that ETL-Span cannot solve the problem of entity pair overlapping. Nevertheless, ETL-Span still surpasses baselines in all categories. Specifically, ETL-Span outperforms OrderRL by 6.1% on the Normal class, 6.9% on the SEO class, and 0.6% 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.

We also compare the results given different numbers of triplets in a sentence, and sentences in the NYT-multi test set are divided into 5 subclasses, each class contains sentences that has 1,2,3,4 or ≥ 5 triplets. As illustrated in Figure 4, ETL-Span outperforms the baselines under all numbers of triplets in a sentence. When the sentence only contains one triplet, ETL-Span yields a 8.8% improvement in comparison with OrderRL. When there are multiple triplets in a sentence, ETL-Span still outperforms GraphRel and OrderRL significantly. These observations demonstrate that our extraction paradigm

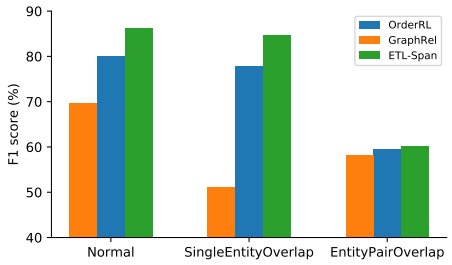

Figure 3. F1 score by overlapping category on the NYT-multi test set.

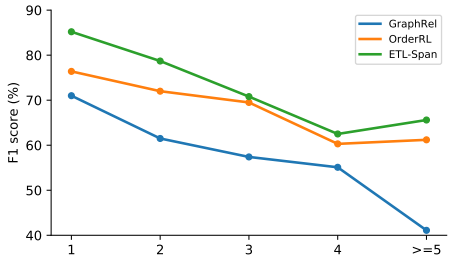

Figure 4. F1 score by sentence triplet count on the NYT-multi test set.

is effective to handle the multiple relation extraction task.

4 RELATED WORK

There have been extensive studies for entity relation extraction task. Traditional pipelined methods divide this task into two separate subtasks: first extract the token spans in the text to detect entity mentions, and then discover the relational structures between entity mentions [28]. Entity recognization has traditionally been solved as a sequence labeling problem, and most recent work leverages a LSTM-CRF architecture [12, 16, 34]. Relation extraction is normally treated as a problem of multi-label classification [36]. Zeng et al. [29] employed a deep convolutional neural network for extracting lexical and sentence level features. Zhou et al. [37] combined attention mechanisms with BiLSTM to reduce intra-sentence noise. Yu et al. [27] proposed to learn the latent relational expressions based on the segment attention layer for relation extraction. However, all these methods require preprocessing step such as NER and ignore interactions between entity recognization and relation extraction, therefore may suffer from error propagation [2, 3, 20, 24].

To address the above limitation, a variety of joint learning methods were proposed [6, 10, 14, 25, 32]. Kate et al. [9] presented a card-pyramid graph to represent entities and their relations in a sentence. Miwa et al. [18] introduced a simple and flexible table representation of entities and relations. However, these models need complicated process of feature engineering, which requires much manual efforts and domain expertise. Recently, several end-to-end neural architectures are applied to joint relation extraction. Sun et al. [24] optimized a global loss function to jointly train entity recognition model and relation classification model under the framework work of Minimum Risk Training. Bekoulis et al. [1, 2] first recognized the entities, then they formulated the relation extraction task as a multi-head selection problem. For each entity, they calculated the score between it and every other entities for a given relation. Tan et al. [26] first identified all candidate entities, then performed relation extraction via ranking with translation mechanism. Sun et al. [23] developed an entity-relation bipartite graph to perform joint inference on entity types and relation types. Fu et al. [4] utilized graph convolutional network to extract overlapping relations by splitting entity mention pairs into several word pairs and considering all pairs for prediction. Nevertheless, these extract-then-classify methods still require explicit separate components for entity extraction and relation classification, and the relation classifier may be overwhelmed by the redundant extracted entity pairs. Another line of work [30, 31] directly generated triplets one by one by a sequence-to-sequence model but fail to extract an entity that has multiple words. Zheng et al. [35] proposed a unified tagging model which utilizes a special tagging scheme to convert joint extraction task to a sequence tagging problem. However, their model cannot recognize overlapping relations in the sentence. As the improvement, Dai et al. [3] proposed to extract overlapping triplets by tagging one n-word sentence for n times. Unfortunately, due to the labeling-once process, this kind of unified labeling methods cannot fully exploit the inter-dependency between entities and relations.

In this paper, we design a novel joint extraction paradigm which first extracts head-entities and then labels tail-entities and relations for each head-entity. In essence, it bridges the gap between extract-then-classify and unified labeling approaches. More specifically, when compared with the extract-then-classify methods, our extract-then-label paradigm no longer extracts all entities at the first step, only head-entities that are likely to participate in target triplets are identified, thus alleviating the impact of redundant entity pairs. Owing to the reasonable decomposition strategy, our model can better capture the correlations between head-entities and tail-entities than unified labeling approaches, thus resulting in a better joint extraction performance. Besides, our span-based tagging scheme is inspired by recent advances in machine reading comprehension [22], which derived the answer by predicting its start and the end indices in the paragraph. Hu et al. [8] also applied this sort of architecture to open-domain aspect extraction and achieved great success.

5 CONCLUSIONS

In this paper, we present an end-to-end sequence labeling framework for joint extraction of entities and relations based on a novel decomposition strategy. 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 public datasets. Further analysis demonstrates the ability of our model in handling normal, overlapping and multiple relation extraction. In the future, we would like to explore similar decomposition strategy in other information extraction tasks, such as event extraction and aspect extraction. The source code of this paper can be obtained from https://github.com/yubowen-ph/JointER.

6 ACKNOWLEDGEMENTS

The authors thank Jianlin Su from Zhuiyi Tech Co. Ltd.for his helpful discussions. The work presented in this paper is supported by the National Key Research and Development Program of China (grant No.2016YFB0801003) and the Strategic Priority Research Program of Chinese Academy of Sciences (grant No.XDC02040400).

REFERENCES

[1] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder, 'Adversarial training for multi-context joint entity and relation extraction', in *Proc. of EMNLP*, pp. 2830–2836, (2018).
[2] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder, 'Joint entity recognition and relation extraction as a multi-head selection problem', *Expert Systems with Applications*, **114**, 34–45, (2018).
[3] Dai Dai, Xinyan Xiao, Yajuan Lyu, Shan Dou, Qiaoqiao She, and Haifeng Wang, 'Joint extraction of entities and overlapping relations using position-attentive sequence labeling', in *Proc. of AAAI*, pp. 6300–6308, (2019).
[4] Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma, 'Graphrel: Modeling text as relational graphs for joint entity and relation extraction', in *Proc. of ACL*, pp. 1409–1418, (2019).
[5] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini, 'Creating training corpora for nlg micro-planners', in *Proc. of ACL*, pp. 179–188, (2017).
[6] Pankaj Gupta, Hinrich Schtze, and Bernt Andrassy, 'Table filling multi-task recurrent neural network for joint entity and relation extraction', in *Proc. of COLING*, pp. 2537–2547, (2016).
[7] Sepp Hochreiter and Jrgen Schmidhuber, 'Long short-term memory', *Neural computation*, 1735–1780, (1997).
[8] Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv, 'Open-domain targeted sentiment analysis via span-based extraction and classification', *arXiv preprint arXiv:1906.03820*, (2019).
[9] Rohit J Kate and Raymond J Mooney, 'Joint entity and relation extraction using card-pyramid parsing', in *Proc. of CONLL*, pp. 203–212. Association for Computational Linguistics, (2010).
[10] Arzoo Katiyar and Claire Cardie, 'Going out on a limb: Joint extraction of entity mentions and relations without dependency trees', in *Proc. of ACL*, pp. 917–928, (2017).
[11] Diederik P Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*, (2014).
[12] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer, 'Neural architectures for named entity recognition', in *Proc. of NAACL*, pp. 260–270, (2016).
[13] Qi Li and Heng Ji, 'Incremental joint extraction of entity mentions and relations', in *Proc. of ACL*, pp. 402–412, (2014).
[14] Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li, 'Entity-relation extraction as multi-turn question answering', *arXiv preprint arXiv:1905.05529*, (2019).
[15] Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han, 'Empower sequence labeling with task-aware neural language model', in *Proc. of AAAI*, (2018).
[16] Xue Mengge, Yu Bowen, Liu Tingwen, Wang Bin, Meng Erli, and Li Quangang, 'Porous lattice-based transformer encoder for chinese ner', *arXiv preprint arXiv:1911.02733*, (2019).
[17] Makoto Miwa and Mohit Bansal, 'End-to-end relation extraction using lstms on sequences and tree structures', *arXiv preprint arXiv:1601.00770*, (2016).
[18] Makoto Miwa and Yutaka Sasaki, 'Modeling joint entity and relation extraction with table representation', in *Proc. of EMNLP*, pp. 1858–1869, (2014).
[19] Jeffrey Pennington, Richard Socher, and Christopher Manning, 'Glove: Global vectors for word representation', in *Proc. of EMNLP*, pp. 1532–1543, (2014).
[20] Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han, 'Cotype: Joint extraction of typed entities and relations with knowledge bases', in *Proc. of WWW*, pp. 1015–1024, (2017).
[21] Sebastian Riedel, Limin Yao, and Andrew McCallum, 'Modeling relations and their mentions without labeled text', in *Proc. of ECML*, pp. 148–163, (2010).
[22] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi, 'Bidirectional attention flow for machine comprehension', *arXiv preprint arXiv:1611.01603*, (2016).
[23] Changzhi Sun, Yeyun Gong, Yuanbin Wu, Ming Gong, Daxin Jiang, Man Lan, and Sun, 'Joint type inference on entities and relations via graph convolutional networks', in *Proc. of ACL*, pp. 1361–1370, (2019).
[24] Changzhi Sun, Yuanbin Wu, Man Lan, Shiliang Sun, Wenting Wang, Kuang-Chih Lee, and Kewen Wu, 'Extracting entities and relations with joint minimum risk training', in *Proc. of EMNLP*, pp. 2256–2265, (2018).
[25] Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang, 'A hierarchical framework for relation extraction with reinforcement learning', in *Proc. of AAAI*, pp. 7072–7079, (2019).
[26] Zhen Tan, Xiang Zhao, Wei Wang, and Weidong Xiao, 'Jointly extracting multiple triplets with multilayer translation constraints', in *Proc. of AAAI*, (2019).
[27] Bowen Yu, Zhenyu Zhang, Tingwen Liu, Bin Wang, Sujian Li, and Quangang Li, 'Beyond word attention: using segment attention in neural relation extraction', in *Proc. of IJCAI*, pp. 5401–5407. AAAI Press, (2019).
[28] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella, 'Kernel methods for relation extraction', *Journal of machine learning research*, 1083–1106, (2003).
[29] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao, 'Relation classification via convolutional deep neural network', in *Proc. of COLING*, pp. 2335–2344, (2014).
[30] Xiangrong Zeng, Shizhu He, Daojian Zeng, Kang Liu, Shengping Liu, and Jun Zhao, 'Learning the extraction order of multiple relational facts in a sentence with reinforcement learning', in *Proc. of EMNLP*, pp. 367–377, (2019).
[31] Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao, 'Extracting relational facts by an end-to-end neural model with copy mechanism', in *Proc. of ACL*, pp. 506–514, (2018).
[32] Meishan Zhang, Yue Zhang, and Guohong Fu, 'End-to-end neural relation extraction with global optimization', in *Proc. of EMNLP*, pp. 1730–1740, (2017).
[33] Xiao Zhang and Dan Goldwasser, 'Sentiment tagging with partial labels using modular architectures', *arXiv preprint arXiv:1906.00534*, (2019).
[34] Yue Zhang and Jie Yang, 'Chinese ner using lattice lstm', in *Proc. of ACL*, pp. 1554–1564, (2018).
[35] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu, 'Joint extraction of entities and relations based on a novel tagging scheme', in *Proc. of ACL*, pp. 1227–1236, (2017).
[36] Zhang Zhenyu, Shu Xiaobo, Yu Bowen, Liu Tingwen, Zhao Jiapeng, Li Quangang, and Guo Li, 'Distilling knowledge from well-informed soft labels for neural relation extraction', in *Proc. of AAAI*, (2020).
[37] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu, 'Attention-based bidirectional long short-term memory networks for relation classification', in *Proc. of ACL*, pp. 207–212, (2016).