Enhanced Language Representation with Label Knowledge for Span Extraction

Pan Yang[1,2]*, **Xin Cong**[3,4], **Zhenyun Sun**[1,5], **Xingwu Liu**[6]†
[1]Institute of Computing Technology, Chinese Academy of Sciences
[2]PCG, Tencent
[3]Institute of Information Engineering, Chinese Academy of Sciences
[4]School of Cyber Security, University of Chinese Academy of Sciences
[5]School of Computer Science and Technology, University of Chinese Academy of Sciences
[6]School of Mathematical Sciences, Dalian University of Technology
im.panyang@gmail.com, congxin@iie.ac.cn
sunzhenyu@ict.ac.cn, liuxingwu@dlut.edu.cn

Abstract

Span extraction, aiming to extract text spans (such as words or phrases) from plain texts, is a fundamental process in Information Extraction. Recent works introduce the label knowledge to enhance the text representation by formalizing the span extraction task into a question answering problem (QA Formalization), which achieves state-of-the-art performance. However, QA Formalization does not fully exploit the label knowledge and suffers from low efficiency in training/inference. To address those problems, we introduce a new paradigm to integrate label knowledge and further propose a novel model to explicitly and efficiently integrate label knowledge into text representations. Specifically, it encodes texts and label annotations independently and then integrates label knowledge into text representation with an elaborate-designed semantics fusion module. We conduct extensive experiments on three typical span extraction tasks: flat NER, nested NER, and event detection. The empirical results show that 1) our method achieves state-of-the-art performance on four benchmarks, and 2) reduces training time and inference time by 76% and 77% on average, respectively, compared with the QA Formalization paradigm. Our code and data are available at https://github.com/Akeepers/LEAR.

1 Introduction

Information Extraction (IE), a fundamental task in natural language processing, aims to extract structured knowledge from unstructured texts. It usually contains the process that extracts text spans (such as words or phrases) from plain text, e.g., NER. Span extraction is usually formulated into the sequence labeling problem that assigns a categorical label to each token in a text.

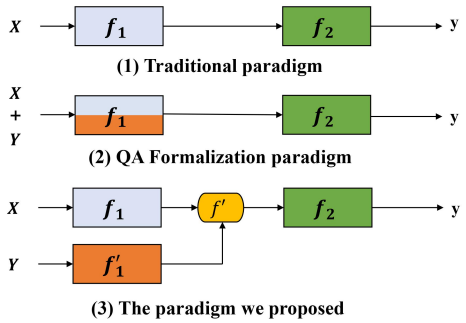

Figure 1: Illustration of different paradigms[1] for span extraction. X represents the sequence; Y represents the category-related extra input (e.g., question in QA paradigm); y represents corresponding category; f_1 is the encoder to learning text representation; f_2 is the task layer to decode results; f'_1 is the extra encoder to learn the representation of Y; f' is the extra module for the fusion of text semantics and label knowledge.

Many efforts have been devoted to span extraction. Early approaches are mainly based on handcrafted features such as domain dictionaries (Sekine and Nobata, 2004; Etzioni et al., 2005) and lexical features (Ahn, 2006). As neural networks show the effectiveness of learning text features automatically, many neural-based methods have been proposed (Huang et al., 2015; Strubell et al., 2017; Liu et al., 2018a; Cui et al., 2020). Recently, self-attention-based pre-trained language models such as BERT (Devlin et al., 2019) are widely used to boost the span extraction task (Devlin et al., 2019; Yang et al., 2019a). However, most existing methods treat labels as independent and meaningless one-hot vectors, neglecting prior information of labels (referred to as label knowl-

*This work was done in ICT, CAS.
†Corresponding Author

[1]The traditional paradigm represents the methods that ignore the label knowledge.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 4623–4635
November 7–11, 2021.

edge).

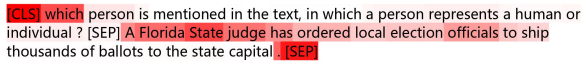

Figure 2: The Visualization of attention mechanism for the token "judge" (QA Formalization).[2] The darker color indicates the higher attention score.

To alleviate the limitation, several studies (Wang et al., 2018; Lin et al., 2019a; Chen et al., 2020) start to integrate label knowledge into span extraction. Among them, QA Formalization is especially attractive due to its effectiveness (Levy et al., 2017; Li et al., 2019, 2020b; Liu et al., 2020; Du and Cardie, 2020; Li et al., 2020a). Simply put, QA Formalization treats span extraction as a question answering problem. Taking NER as an example, to extract "PERSON" entities, it is formalized as answering the question *"which person is mentioned in the text, in which a person represents a human or individual?"* based on the given text. Benefiting from the label knowledge of the category-related questions, QA Formalization usually yields state-of-the-art performance in span extraction even in low-resource scenarios.

QA Formalization, however, exhibits two key weaknesses: 1) **Inefficiency**: Formalizing the span extraction as QA causes a drastic reduction of training/inference efficiency. Specifically, the typical QA Formalization method concatenates *question* and *text* as the input (e.g., [CLS] *question* [SEP] *text* [SEP]) and jointly encodes *question* and *text* with a transformer-based encoder. The joint-encoding has to transform every *text* into $|C|$ pairs of the form $\langle question, text \rangle$, where $|C|$ is the size of the label category set. This transformation, which increases both the size of the sample set and the length of text sequences, finally increases the time cost of training and inference. 2) **Underutilization**: The label knowledge is integrated implicitly into text representation based on the self-attention mechanism (Vaswani et al., 2017). As Figure 2 shows, the "attention" of self-attention mechanism will be distracted by *text*, not entirely focus on the *question* part. Thus, the label knowledge is not fully exploited to enhance the text representations.

To address aforementioned two problems, we propose a novel paradigm (seen in Figure 1) to integrate label knowledge. First, since joint-encoding causes low efficiency, we decompose *question-text* encoding process into two separate encoding modules: the *text* encoding module f_1 and the *question* encoding module f_1'. In this way, the size of the sample set is no longer expanded by $|C|$ times. Second, to fully utilize the label knowledge, a fusion module f' is designed to explicitly integrate the label and the text representations.

To instantiates the above paradigm, we further propose a model termed as **LEAR** to learn Label-knowledge EnhAnced Representation. A powerful encoder f_1' is essential for understanding the *label annotations*[3]. However, training the encoder f_1' from scratch is challenging since the number of *label annotations* is too small. Thus we share the weights of f_1 and f_1' (called *shared encoder*), which can learn the label knowledge by large pre-trained model and does not introduce extra parameters. Next, the learned label knowledge is integrated into text representations by the semantics-guided attention module. We conduct experiments in five benchmarks on three typical span extraction tasks: flat NER, nested NER, and event detection (ED). Compared with QA Formalization baselines, our model LEAR outperforms them to achieve a new state-of-the-art. Furthermore, LEAR reduces training time and inference time by 76% and 77% on average, respectively.

To sum up, our contributions are as follows:

- We propose a new paradigm to exploit label knowledge to boost span extraction, which encodes *texts* and *label annotations* independently and integrates label knowledge into text representation explicitly.
- We propose a novel model, LEAR, to instantiate the above paradigm. It designs the *shared encoder* and semantics-guided attention to tackle the technical challenges.
- The experiments show that our method achieves SOTA performance on four benchmarks, and it is much faster than the previous SOTA approach. Further analysis confirms the effectiveness and efficiency of our model.

2 Preliminaries

2.1 Task Formalization

We formulate the following span extraction task: given an input text $X = (x_1, x_2, \cdots, x_n)$ consisting of n tokens, find out all candidate spans in X

[2]The attention score comes from the well-trained model based on the pervious work (Li et al., 2020b).

[3]Previous SOTA QA Formalization method (Li et al., 2020b) adopts the annotations of label as *questions*.

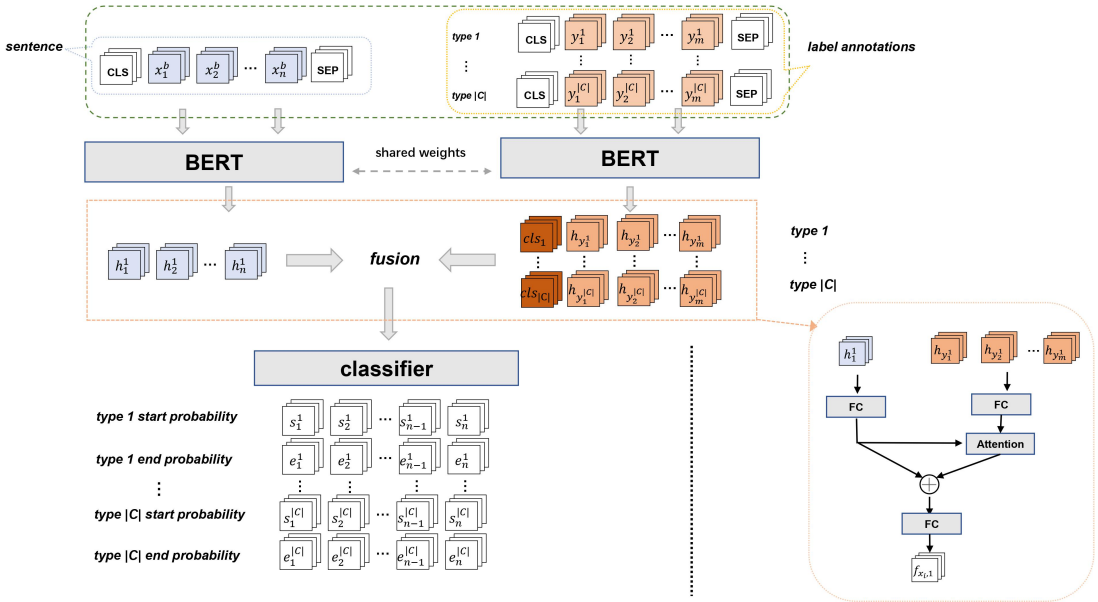

Figure 3: Illustration of our LEAR.

and assign a label $c \in C$ to each of them, where C is a predefined set of categories (or tag types, interchangeably).

This formulation provides a uniform framework for modeling many important problems. For example, when C is the set of event types such as die, attack, marry, and so on, span extraction is exactly the event detection task. In addition, if C consists of entity types such as persons, organizations, locations, span extraction turns into the well-known named entity recognition task.

2.2 Data Construction

Task	Category	Label Annotation
ED	Die	a die Event occurs whenever the life of a person entity ends.
NER	Person	a person entity is limited to human including a single individual or a group.

Table 1: Label categories and their corresponding annotations.

QA formalization is powerful in span extraction since it incorporates label knowledge. One of its prerequisites is the existence of reasonable *questions*. Usually, *questions* are generated by a manually-designed pre-processing step, which is costly and lacks versatility and accessibility. For instance, Du and Cardie (2020) and Li et al. (2020b) use a purpose-designed template to generate questions, while Liu et al. (2020) exploits a well-designed large pretraining model.

Previous work[4] (Li et al., 2020b) on flat and nested NER uses the annotations of each category (referred to as *label annotations*) as the *questions*. We follow this setting in our work for a fair comparison. Similarly, we utilize the annotations of event types in ACE 2005 event detection task[5]. Table 1 presents an example of those annotations.

3 Approach

In this section, we first give an overall description of our LEAR architecture. LEAR consists of three crucial modules: semantics encoding module, semantics fusion module, and span decoding module. Our architecture (Figure 3) takes text X and label annotation Y of category set C as input. The two inputs are respectively processed by two encoder networks whose backbone is BERT (Devlin et al., 2019). The two encoders share weights (referred to as *shared encoder*) while processing the two inputs. Then the text embedding and label embedding produced by the *shared encoder* are fused by the semantic fusion module to derive the label-knowledge-enhanced embeddings for the text. Finally, the label-knowledge-enhanced embeddings are used to predict whether or not each token is a start or end index for some category.

[4] *Questions* are available in their open source project.

[5] All label annotations are available at: https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf.

3.1 Semantics Encoding Module

Semantics encoding module aims to encode the text and the label annotation into real-valued embeddings. Since the number of label annotations is small compared with the whole sample set, it is challenging to build an encoder from scratch for the label annotations. Thus we introduce the shared encoder, which is inspired by siamese networks (Bromley et al., 1993). The shared encoder is efficient in learning the representation of label annotations and does not introduce extra parameters.

Given input text X and label annotations Y, LEAR first extracts their embeddings $h_X \in \mathcal{R}^{n\times d}$ and $h_Y \in \mathcal{R}^{|C|\times m\times d}$, where n is the length of X, m is the length of label annotation, $|C|$ is the size of the category set C, and d is the vector dimension of the encoder. We denote this operation as:

$$h_X = f_1(X) \tag{1}$$
$$h_Y = f_1'(Y) \tag{2}$$

3.2 Semantic Fusion

The semantic fusion module aims at enhancing the text representation with label knowledge explicitly. To this end, we devise a semantics-guided attention mechanism to achieve this goal.

Specifically, we first feed h_X and h_Y into a fully connected layer, respectively, to map their representations into the same feature space:

$$h'_X = \mathbf{U}_1 \cdot h_X \tag{3}$$
$$h'_Y = \mathbf{U}_2 \cdot h_Y \tag{4}$$

where $\mathbf{U}_1, \mathbf{U}_2 \in \mathcal{R}^{d\times d}$ be the learnable parameters of the fully connected layers.

Then, we apply the attention mechanism over the label annotations for each token in the text. For any $1 \le i \le n$, let x_i be the ith token of X, and $h'_{x_i} \in \mathcal{R}^d$ be the ith row of h'_X. Likewise, for any $1 \le i \le m$ and category $c \in C$, let y_j^c be the jth token of the annotation of c, and $h'_{y_j^c}$ be its embedding from h'_Y. We compute the dot product of h'_{x_i} and $h'_{y_j^c}$, and apply a softmax function to obtain the attention scores:

$$a_{x_i,y_j^c} = \frac{\exp\left(h'_{x_i} \cdot h'_{y_j^c}\right)}{\sum_j \exp\left(h'_{x_i} \cdot h'_{y_j^c}\right)} \tag{5}$$

Finally, we get the fine-grained features by attention, which is in turn fused into token embedding by *add* operation:

$$h^c_{x_i} = h'_{x_i} + \sum_j a_{x_i,y_j^c} \cdot h'_{y_j^c} \tag{6}$$
$$\hat{h}^c_{x_i} = \tanh\left(\mathbf{V} \cdot h^c_{x_i} + \mathbf{b}\right) \tag{7}$$

where $\tanh(\cdot)$ is the hyperbolic tangent function, and $\mathbf{V} \in \mathcal{R}^{d\times d}$ and $\mathbf{b} \in \mathcal{R}^d$ are learnable parameters. Intuitively, $\hat{h}^c_{x_i}$ encodes the information related to category c.

Repeating the process for all categories, we obtain the category-related embedding $\hat{h}_{x_i} = (\hat{h}^1_{x_i}, \cdots, \hat{h}^{|C|}_{x_i})$ for each token x_i.

3.3 Span Decoding

Now we are ready to select spans. Following Li et al. (2020b), we use the start/end tagging schema to annotate the target spans to extract. Specifically, for each token x_i, we compute the following vector:

$$\text{start}_{x_i} = \text{sigmoid}(f_o(\mathbf{M_s} \circ \hat{h}_{x_i} + \mathbf{b_s})) \tag{8}$$

where $\mathbf{M_s} \in \mathcal{R}^{|C|\times d}$ and $\mathbf{b_s} \in \mathcal{R}^d$ are learnable parameters, $\circ$ is the element-wise multiplication, and $f_o(\cdot)$ is the function that sums up the rows of the input matrix. Intuitively, for any $c \in C$, $\text{start}^c_{x_i}$ indicates the probability that x_i starts a span of the category c.

Likewise, we obtain the end_{x_i}, which indicates the probabilities that x_i ends a span, in the same prediction procedure. Then we extract the results case by case, depending on whether or not spans of the same category can be nested[6].

Flat Span Decoding This is the case without nested spans in the same category.

The most widely adopted method is the *nearest matching principle* (Du and Cardie, 2020; Wei et al., 2020), which matches a start position of category c with the nearest next end position of c.

In contrast, we follow the *heuristic matching principle* (Yang et al., 2019b), which determines spans from the lens of probability. Roughly speaking, among candidate start and end positions of a category c, we only match those having high probabilities, where the probabilities are derived from vectors defined in formulas (8) For detailed information of heuristic matching, please refer to the algorithm in Appendix A.1.

The two principles for span decoding are further compared by experiments in Appendix A.2.

[6] *Nested* here represents both nested and overlapped spans, just like nested NER (Finkel and Manning, 2009).

Nested Span Decoding Now suppose that spans in the same category may be nested or overlapped.

Since the *heuristic matching principle* does not work anymore, we follow the solution of BERT-MRC (Li et al., 2020b). It employs a binary classifier to predict the probability that a pair of candidate start/end positions should be matched as a span. Specifically, for any category c, define the following binary classifier:

$$P^c_{i,j} = \text{sigmoid}(\mathbf{M} \cdot \text{concat}(\hat{h}^c_{x_i}, \hat{h}^c_{x_j})) \quad (9)$$

where $1 \leq i, j \leq n$, and $\mathbf{M} \in \mathcal{R}^{1\times 2d}$ is the learnable parameter. When $P^c_{i,j} > 0.5$, it will be predicted that x_i and x_j demarcate a span of c.

3.4 Loss Function

Given input text $X = (x_1, x_2, \cdots, x_n)$ consisting of n tokens and set C of categories, for any $c \in C$, define $S^c \in \{0, 1\}^n$ to be the vector whose ith entry $S^c_{x_i} = 1$ if and only if x_i is a ground-truth start position of c. Likewise, define $E^c \in \{0, 1\}^n$ to indicate the ground-truth end positions. Recall the vectors start^c and end^c defined in Section 3.3. Define start loss function $\mathcal{L}_s$ and end loss function $\mathcal{L}_e$ of our model as follows:

$$\mathcal{L}_s = \frac{1}{n} \sum_{c\in C} \sum_{1\leq i\leq n} \text{CE}\left(\text{start}^c_{x_i}, S^c_{x_i}\right)$$

$$\mathcal{L}_e = \frac{1}{n} \sum_{c\in C} \sum_{1\leq i\leq n} \text{CE}\left(\text{end}^c_{x_i}, E^c_{x_i}\right)$$

where CE stands for the cross entropy.

Flat Span Extraction The final loss function of our model is defined to be $\mathcal{L} = \mathcal{L}_s + \mathcal{L}_e$.

Nested Span Extraction More notation is needed. Recall the matrix $P^c \in \mathcal{R}^{n\times n}$ defined in Formula (9). Let $M^c \in \mathcal{R}^{n\times n}$ be the binary matrix such that $M^c_{i,j} = 1$ if and only if the tokens x_i and x_j demarcate a ground-truth span of category c. Define the match loss function

$$\mathcal{L}_{match} = \frac{1}{n^2} \sum_{1\leq i,j\leq n} \sum_{c\in C} \text{CE}\left(P^c_{i,j}, M^c_{i,j}\right) W^c_{i,j}$$

where $W^c \in \mathcal{R}^{n\times n}$ is the binary matrix such that $W^c_{i,i} = 1$ if and only if $P^c_{i,j} > 0.5$ or $M^c_{i,j} = 1$.

Then the final loss function of our model is defined to be $\mathcal{L} = \alpha(\mathcal{L}_s + \mathcal{L}_e) + \beta\mathcal{L}_{match}$, where α, β are hyper-parameters to control the contributions towards the overall training objective.

4 Experiments

In this section, we present LEAR results on 5 widely-used benchmarks.

4.1 Datasets

Dataset We evaluate our model on three span extraction tasks: flat NER, nested NER and event detection. For flat NER, we conduct experiments on MSRA (Levow, 2006) and Chinese OntoNote 4.0 (Pradhan et al., 2011). For nested NER, we evaluate our model on ACE 2004 (Doddington et al., 2004) and ACE 2005 (NER) datasets. For event detection, we use the ACE 2005[7] (ED) dataset.

For MSRA and Chinese OntoNote 4.0, which contains three and four types of entities respectively, we follow the data preprocessing strategies in Li et al. (2020b) and Meng et al. (2019) for fair comparison. ACE 2005 (NER) and ACE 2004 both annotate 7 entity categories. For ACE 2005 (NER), we use the same data split as previous works (Lin et al., 2019b); for ACE 2004, We use the same setup as Katiyar and Cardie (2018). ACE 2005 (ED) annotates 33 types of events and we follow the same settings of Chen et al. (2015) and Chen et al. (2018) to split data into train, development, and test set. More statistics of datasets are listed in Appendix A.4.

4.2 Baselines

Named Entity Recognition We use the following models as baselines: (1) **BiLSTM-CRF** (Ma and Hovy, 2016) uses the Bi-LSTM layer as encoder. (2) **Seg-Graph** (Wang and Lu, 2018) proposes a segmental hypergragh representation to model overlapping entity mentions. (3) **BERT-Tagger** (Devlin et al., 2019) treats NER as a tagging task with a bidirectional encoder representations. (4) **Lattice-LSTM** (Zhang and Yang, 2018) constructs a word-character lattice for Chinese NER. (5) **Glyce-BERT** (Meng et al., 2019) combines glyph information with BERT pretraining for Chinese NER. (6) **Seq2Seq-BERT** (Shibuya and Hovy, 2020) views the nested NER as a sequence-to-sequence problem. (7) **Biaffine-NER** (Yu et al., 2020) predicts named entity with a biaffine network. (8) **BERT-MRC** (Li et al., 2020b) treats NER as a MRC/QA task, which is the state-of-the-art method on both flat and nested NER.

[7]This corpora is designed for multi-tasks, such as event detection and NER. Data source: https://catalog.ldc.upenn.edu/LDC2006T06

Event Detection We compare with the following methods: (1) **DMCNN** (Chen et al., 2015) builds a dynamic multi-pooling convolutional model; (2) **JRNN** (Nguyen et al., 2016) employs bidirectional RNN for ED; (3) **ANN-AugAtt** (Liu et al., 2017) uses annotated event argument information to get better attention scores; (4) **JMEE** (Liu et al., 2018b) enhances GCN with self-attention and highway network; (5) **EE-GCN** (Cui et al., 2020) learns token representation via edge-enhanced GCN with specific syntactic label incorporated. (6) **EKD** (Tong et al., 2020) is the state-of-the-art method on the ACE2005 dataset. (7) **BERT_QA_Trigger** (Du and Cardie, 2020) formalizes event detection as a QA task.

Furthermore, to compare the efficiency between QA Formalization and LEAR, we instantiate the traditional paradigm as a baseline for efficiency comparison in the simplest way, which only contains a BERT encoder and two fully connected layers as the classifiers. We denote this baseline model as **Traditional Formalization**.

4.3 Experimental Setups

We use BERT (Devlin et al., 2019) as the backbone to learning the contextualized representation of the texts. More specifically, we implement our model based on the BERT-large model for NER task, which is the same as BERT-MRC (Li et al., 2020b). In the event detection task, we use the BERT-base model as the backbone. We adopt the adam optimizer (Kingma and Ba, 2015) with a linear decaying schedule to train our model. The detail of hyper-parameters settings is listed in Appendix A.3.

To make results comparable in the efficiency comparison experiment (as shown in Table 3), all models take the BERT-base as the backbone and set all hyperparameters to the same except *max_seq_len* of QA Formalization. The higher *max_seq_len* meets the requirement of taking the *question* as extra input for QA Formalization.

Effectiveness Evaluation We use micro-average precision, recall, and F1 as evaluation metrics. A prediction is considered correct only if both its boundary and category are predicted correctly.

Efficiency Evaluation We use the time costs (in seconds) of training and inference to evaluate the efficiency of different models. Specifically, 1) *Training*: the time cost of training in one epoch; 2) *Inference*: the time cost for the model to get all prediction results of the test set.

English ACE2005 for ED (Flat)			
Model	**P**	**R**	**F1**
DMCNN (Chen et al., 2015)	75.6	63.6	69.1
JRNN (Nguyen et al., 2016)	66.0	73.0	69.3
ANN-AugAtt (Liu et al., 2017)	78.0	66.3	71.7
JMEE (Liu et al., 2018b)	76.3	71.3	73.7
EE-GCN (Cui et al., 2020)	76.7	78.6	77.6
BERT_QA_Trigger† (Du and Cardie, 2020)	71.12	73.70	72.39
EKD (Tong et al., 2020)	79.1	78.0	78.6
LEAR	**82.04**	**81.18**	**81.61**
English ACE 2004 for NER (Nested)			
Model	**P**	**R**	**F1**
Seg-Graph (Wang and Lu, 2018)	78.0	72.4	75.1
Seq2seq-BERT (Straková et al., 2019)	-	-	84.40
DYGIE (Luan et al., 2019)	-	-	84.7
BERT-MRC† (Li et al., 2020b)	85.05	**86.32**	85.98
Biaffine-NER (Yu et al., 2020)	87.3	86.0	86.7
LEAR	**87.89**	85.86	**86.87**
English ACE 2005 for NER (Nested)			
Model	**P**	**R**	**F1**
Seg-Graph (Wang and Lu, 2018)	76.8	72.3	74.5
DYGIE (Luan et al., 2019)	-	-	82.9
Seq2seq-BERT (Straková et al., 2019)	-	-	84.33
Biaffine-NER (Yu et al., 2020)	85.2	85.6	85.4
BERT-MRC† (Li et al., 2020b)	**87.16**	86.59	**86.88**
LEAR	84.85	**87.95**	86.63
Chinese OntoNotes 4.0 for NER (Flat)			
Model	**P**	**R**	**F1**
Lattice-LSTM (Zhang and Yang, 2018)	76.35	71.56	73.88
BERT-Tagger (Devlin et al., 2019)	78.01	80.35	79.16
Glyce-BERT (Meng et al., 2019)	81.87	81.40	81.63
BERT-MRC† (Li et al., 2020b)	**82.98**	81.25	82.11
LEAR	81.12	**84.86**	**82.95**
Chinese MSRA for NER (Flat)			
Model	**P**	**R**	**F1**
Lattice-LSTM (Zhang and Yang, 2018)	93.57	92.79	93.18
BERT-Tagger (Devlin et al., 2019)	94.97	94.62	94.80
Glyce-BERT (Meng et al., 2019)	95.57	95.51	95.54
BERT-MRC† (Li et al., 2020b)	96.18	95.12	95.75
LEAR	**96.23**	**95.57**	**95.96**

Table 2: Results in five benchmarks. The best results are in **bold**, † means QA Formalization methods.

4.4 Main Results

Effectiveness Table 2 shows the performance of our LEAR compared with the above state-of-the-art methods on the test sets. We can see that our LEAR outperforms all other models on four benchmarks, i.e., +3.01%, +0.84%, +0.21%, +0.17%, respectively on ACE 2005 (ED), OntoNote 4.0, MSRA and ACE 2004. This improvement indicates that the *explicit* fusion with a dedicated module is better than the *implicit* fusion based on the self-attention mechanism. Since the joint-encoding of QA Formalization, the "attention" of self-attention mechanism will be distracted by *text*, not entirely focus on the *question*. Thus the label knowledge introduced by *label annotation* is not fully exploited. By con-

Task	Dataset	|C|	Traditional Formalization		QA Formalization		LEAR	
			Train	Inference	Train	Inference	Train	Inference
ED	ACE 2005	33	349.9(x1.0)	5.8(x1.0)	11456.2(x32.7)	176.1(x30.4)	1005.5(x2.9)	9.8(x1.7)
NER	MSRA	3	167.8(x1.0)	5.1(x1.0)	626.7(x3.7)	14.5(x2.8)	206.8(x1.2)	5.6(x1.1)
	OntoNotes 4.0	4	58.3(x1.0)	5.2(x1.0)	258.1(x4.4)	19.7(x3.8)	74.4(x1.3)	6.1(x1.2)
	ACE 2005	7	103.9(x1.0)	4.3(x1.0)	684.4(x6.6)	26.3(x6.1)	167.8(x1.6)	5.1(x1.2)
	ACE 2004	7	87.1(x1.0)	3.4(x1.0)	604.8(x6.9)	20.5(x6.0)	145.5(x1.7)	4.3(x1.3)

Table 3: The efficiency comparison of different methods. $(\cdot)$ indicates the relative efficiency compare with the Traditional Formalization (e.g., $\frac{T_{LEAR}}{T_{Traditional\ Formalization}}$).

trast, our LEAR learns knowledge-enhanced representations for each token by a semantics-guided fusion module, whose attention entirely focuses on the *label annotation*.

Efficiency Table 3 shows that our LEAR is much faster than QA Formalization, i.e., reducing the training and inference time by 76% and 77% on average, respectively. The reduction in training/inference time is positively correlated with the number of categories $|C|$, which benefits from breaking the joint-encoding limitation of QA Formalization. As Table 4 shows, the time complexity of LEAR during inference is $\mathcal{O}(n^2 + |C|mn)$, in which we ignore the cost for the encoding of label annotations in our LEAR. Because LEAR only encodes all label annotations once and reuses their representations during the inference, which is favorable for industrial applications in the resource-limited online environment. In contrast, the time complexity of QA Formalization is $\mathcal{O}(|C| \cdot (n + m)^2)$, causing a dramatic decrease in efficiency of inference.

Method	Time Complexity
Traditional Formalization	$\mathcal{O}(n^2)$
QA Formalization	$\mathcal{O}(\|C\| \cdot (n+m)^2)$
LEAR	$\mathcal{O}(n^2 + \|C\|mn)$

Table 4: The time complexity of different model architectures during inference.

To summarize, the fundamental starting points of the proposed paradigm include: 1) decomposing question-text joint encoding into two separate encoding modules; 2) explicitly integrating label knowledge by a dedicated module. The above experiments confirm that our LEAR, an instantiation of the proposed paradigm, outperforms previous SOTA methods in **effectiveness and efficiency**.

5 Analysis

5.1 Analysis for Model Variants

To demonstrate the effectiveness of our method, we build a series of variants of LEAR. For the semantics encoding module, we set: 1) **Label Embedding Layer (LEL)**: replacing the encoder module of label annotations with a label embedding layer, which is initialized by glove (Pennington et al., 2014). The F1 scores drop 0.86% on average. The results show that the improvement of our LERA comes from understanding the label annotation, which is handled well by the shared encoder. 2) **Label Name Encoding (LNE)**: replacing the label annotations with corresponding label names. The results drop 0.53% on average, indicating that label names contain less label knowledge than label annotation.

In order to survey the semantics fusion strategy, we set: 1) **Average Pooling & Add (AP & Add)**: replacing the semantics-guided attention mechanism with average pooling and integrating label knowledge by *add* operation. The F1 scores drop by 0.80% on average. 2) **Sentence Features & Similarity (SF & Sim)**: using the sentence-level features of label annotations (i.e., the embedding of [CLS] symbol) instead of token-level features. Thus the semantics-guided attention mechanism turns into the similarity calculation between token embedding and label feature. The F1 scores drop by 0.56%. The above two settings retain the extra learnable parameters introduced by the fusion module. The results show that the improvement comes from the better exploitation of label knowledge, not the larger parameters. Besides, the results demonstrate that fine-grained (i.e., token-level) features are more effective.

All the above experiments show the effectiveness of our LEAR. Furthermore, the worst-performing variants of LERA still rival the QA Formalization method, which powerfully demonstrates the superiority of the proposed paradigm.

Model	ACE 2005 (ED)	ACE 2005 (NER)	ACE 2004	OntoNotes 4.0	MSRA
LEAR	**81.61**	**86.63**	**86.87**	**82.95**	**95.96**
– LNE	80.96 (↓ 0.65)	85.83 (↓ 0.80)	86.20 (↓ 0.67)	82.58 (↓ 0.37)	95.82 (↓ 0.14)
– LEL	79.72 (↓ 1.89)	85.34 (↓ 1.29)	85.88 (↓ 0.99)	82.92 (↓ 0.03)	95.85 (↓ 0.11)
– AP & Add	79.68 (↓ 1.93)	85.74 (↓ 0.89)	86.18 (↓ 0.69)	82.56 (↓ 0.39)	95.88 (↓ 0.08)
– SF & Sim	79.76 (↓ 1.85)	86.31 (↓ 0.32)	86.67 (↓ 0.2)	82.72 (↓ 0.23)	95.76 (↓ 0.2)

Table 5: The performance of the model variants. The values in table are F1 scores on test sets.

5.2 Performance in Data-Scarce Scenarios

Dataset	Settings	$\text{LEAR}_{\text{w/o}}$	LEAR
ACE 2005 (NER)	1-shot	3.23	**15.42**
	5-shot	38.77	**43.92**
ACE 2004	1-shot	13.30	**22.81**
	5-shot	38.11	**39.03**
OntoNotes 4.0	1-shot	1.89	**7.28**
	5-shot	39.21	**41.32**
MSRA	1-shot	0.16	**0.39**
	5-shot	21.28	**26.22**
ACE 2005 (ED)	1-shot	23.31	**30.23**
	5-shot	63.04	**63.52**

Table 6: F1 scores on exploring the extremely data-scarce scenarios.[8] Both methods take the BERT-base as the base model. The best results are in **bold**.

To verify that exploiting label knowledge is beneficial in data-scarce scenarios, we introduce $\text{LEAR}_{\text{w/o}}$ for comparison. $\text{LEAR}_{\text{w/o}}$ is short for LEAR without label knowledge, whose settings are the same with LEAR except that BERT alone rather than shared encoder and label semantic fusion module are used (i.e., the standard fine-tuning). We conduct two sets of experiments for each dataset using various proportions of the training data: 1-shot and 5-shot. For the 1-shot setting, we sample one sentence for each category in the training set, and the setting of 5-shot is similar. We repeat each experiment 5 times. Tabel 6 shows that our LEAR demonstrates superior performance, for example, obtaining up to +12% absolute improvement and +6.8% on average across all datasets in the 1-shot setting. This is in line with our expectation since LEAR enhances the text representation with label knowledge, which provides more prior information.

In the appendix, we list the further analysis about the effect of different span decoding strategies and the comparison between solving span extraction in the multi-label classification (our LEAR) or sequence-labeling manner (e.g., a CRF layer).

[8] We does not compare with QA paradigm methods because prior works does not report their training data.

6 Related Work

Event Detection (ED). Event Detection aims at extracting event triggers from a text and classifying them. It is dominantly solved in a representation-based manner, where triggers are represented by embedding. In case of no extra information, the representation can be obtained by a powerful text encoder which is usually based on CNN (Chen et al., 2015), RNN (Nguyen et al., 2016), or attention mechanism (Yang et al., 2019b; Tong et al., 2020). Besides, the representation can be enhanced by extra information. Examples of typical extra information include syntactic information (Liu et al., 2018b; Cui et al., 2020) and knowledge base (Liu et al., 2016; Chen et al., 2017). In particular, label knowledge is attracting more and more attention (Li et al., 2020a; Du and Cardie, 2020), which usually formalizes ED as a QA problem.

Named Entity Recognition (NER). Named entity recognition seeks to locate named entities in an unstructured text and classify them into pre-defined categories such as person, organization, location, etc. Traditional methods treat it as a classification task and use CRFs (Lafferty et al., 2001; Sutton et al., 2007) as the backbone. Then neural networks become a prevalent tool in NER with the development of deep learning. Recently, the performance of NER has been further improved by large-scale language models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). When label knowledge is available, state-of-the-art performance can be obtained by formulating NER as a QA problem.

7 Conclusion

In this paper, we propose a novel paradigm to exploit label knowledge to boost the span extraction task and further instantiate a model named LEAR. Unlike the existing QA Formalization methods, LEAR first encodes the text and label annotations independently, and uses a semantic fusion module to integrate label knowledge into the text representation explicitly. In this way, we can overcome the

inefficiency and *underutilization* problems of QA Formalization. Experimental results show that our model outperforms the previous works and enjoys a significantly faster training/inference speed.

Acknowledgments

We would like to thank all reviewers for their insightful comments and suggestions. This work is partially supported by Key-Area Research and Development Program of Guangdong Province (NO.2020B010164003), the National Natural Science Foundation of China (62072433,62090020), the Fundamental Research Funds for the Central Universities (No. DUT21LAB302), Youth Innovation Promotion Association of Chinese Academy of Sciences (2013073), and the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDC05030200).

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–419, Vancouver, Canada. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1267–1276, Brussels, Belgium. Association for Computational Linguistics.

Yunmo Chen, Tongfei Chen, Seth Ebner, Aaron Steven White, and Benjamin Van Durme. 2020. Reading the manual: Event extraction as definition comprehension. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, pages 74–83, Online. Association for Computational Linguistics.

Shiyao Cui, Bowen Yu, Tingwen Liu, Zhenyu Zhang, Xuebin Wang, and Jinqiao Shi. 2020. Edge-enhanced graph convolution networks for event detection with syntactic relation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2329–2339, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 141–150, Singapore. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Gina-Anne Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117, Sydney, Australia. Association for Computational Linguistics.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020a. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838, Online. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020b. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1340–1350, Florence, Italy. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019a. Cost-sensitive regularization for label confusion-aware event detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5278–5283, Florence, Italy. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019b. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5182–5192, Florence, Italy. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online. Association for Computational Linguistics.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging FrameNet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2134–2143, Berlin, Germany. Association for Computational Linguistics.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Vancouver, Canada. Association for Computational Linguistics.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018a. Jointly multiple events extraction via attention-based graph information aggregation. *arXiv preprint arXiv:1809.09078*.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018b. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. 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)*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. 2019. Glyce: Glyph-vectors for chinese character representations. In *Advances in Neural Information Processing Systems*, pages 2746–2757.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 shared task: Modeling unrestricted coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland, Oregon, USA. Association for Computational Linguistics.

Satoshi Sekine and Chikashi Nobata. 2004. Definition, dictionaries and tagger for extended named entity hierarchy. In *LREC*. Lisbon, Portugal.

Takashi Shibuya and Eduard Hovy. 2020. Nested named entity recognition via second-best sequence learning and decoding. *Transactions of the Association for Computational Linguistics*, 8:605–620.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. *arXiv preprint arXiv:1702.02098*.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8(Mar):693–723.

Meihan Tong, Bin Xu, Shuai Wang, Yixin Cao, Lei Hou, Juanzi Li, and Jun Xie. 2020. Improving event detection via open-domain trigger knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5887–5897, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 204–214, Brussels, Belgium. Association for Computational Linguistics.

Zhenghui Wang, Yanru Qu, Liheng Chen, Jian Shen, Weinan Zhang, Shaodian Zhang, Yimei Gao, Gen Gu, Ken Chen, and Yong Yu. 2018. Label-aware double transfer learning for cross-specialty medical named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1–15, New Orleans, Louisiana. Association for Computational Linguistics.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1476–1488, Online. Association for Computational Linguistics.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019a. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–5294, Florence, Italy. Association for Computational Linguistics.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019b. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–5294, Florence, Italy. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Yue Zhang and Jie Yang. 2018. Chinese NER using lattice LSTM. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564, Melbourne, Australia. Association for Computational Linguistics.

A Appendix

A.1 Details of the Heuristic Match Principle

Algorithm 1 span determination (Yang et al., 2019a)

In: start$_c$, end$_c$, and sequence length l.

Out: Result list L that each item is a span plays c^{th} category.

Initiate: $a_s \leftarrow -1$, $a_e \leftarrow -1$
for $i \leftarrow 0$ to l **do**
  if In state 1 **and** $\text{start}^c_{x_i} > 0.5$ **then**
    $a_s \leftarrow i$ and change to state 2
  end if
  if In state 2 **then**
    if the $\text{start}^c_{x_i} > 0.5$ **then**
      $a_s \leftarrow i$ **if** $\text{start}^c_{x_i} > \text{start}^c_{a_s}$
    end if
    if $\text{end}^c_{x_i} > 0.5$ **then**
      $a_e \leftarrow i$ and change to state 3
    end if
  end if
  if In state 3 **then**
    if $\text{end}^c_{x_i} > 0.5$ **then**
      $a_e \leftarrow i$ **if** $\text{end}^c_{x_i} > \text{end}^c_{a_e}$
    end if
    if $\text{start}^c_{x_i} > 0.5$ **then**
      Append $[a_s, a_e]$ to L
      $a_s \leftarrow -1$, $a_e \leftarrow i$ and change to state 2
    end if
  end if
end for

Algorithm 1 contains a finite state machine, which changes from one state to another in response to start^c, end^c. There are three states totally: 1) Neither start nor end has been detected; 2) Only a start has been detected; 3) A start as well as an end have been detected. Specially, the state changes according to the following rules: State 1 changes to State 2 when the current token is a start; State 2 changes to State 3 when the current token is an end; State 3 changes to State 2 when the current token is a new start. Notably, if there has been a start and another start arises, we will choose the one with higher probability, and the same for end.

A.2 Effect of Span Decoding Strategy

Table 7 shows the effect of the different span decoding strategies. All of them use the BERT encoder as backbone. The differences are (1) Strategy **A** treats span decoding as a multi-label classification problem with $2 \times |C|$ binary classifiers, which aims to predict the boundary of a span. This strategy is inspired by the QA task and it is adopted in BERT-span and our LEAR. BERT-span$_{v1}$ employs the heuristic match principle, and BERT-span$_{v2}$ uses the nearest match principle, both mentioned in section 3.3. (2) The most commonly-used Strategy **B** treats span decoding as a multi-class classification problem with BIO or BIOS schema, and is adopted in BERT-softmax and BERT-crf. Compared with BERT-softmax, BERT-crf adds a conditional random field (CRF) layer to model the dependencies between predictions, usually yielding better performance but worse efficiency.

OntoNotes 4.0 for NER		
Strategy	**Method**	**F1**
A	BERT-span$_{v1}$	82.65
	BERT-span$_{v2}$	82.14
B	BERT-crf	81.65
	BERT-softmax	81.30

Table 7: Results with different span decoding strategies. BERT-span$_{v1}$ is the LEAR$_{\text{w/o}}$ mentioned above.

The results show that: (1) The strategy used by LEAR has better performance than the traditional way. The reason might be that, the span decoding strategy in our approach is start/end position matching, which only needs to predict the span's boundary. In contrast, the strategy adopted in previous methods needs to predict both boundary and internal words, which is much harder, especially for a longer span. (2) The comparison between BERT-span$_{v1}$ and BERT-span$_{v2}$ shows that, the heuristic match principle could achieve better results by making the most of information from probability. (3) Besides, there is an extra benefit for Strategy **A**. It naturally tackles the nested span issue, which means that candidate span overlaps with different categories.

A.3 Details of Hyper-Parameters Settings

All hyper-parameters of our model are listed in Table 8 in detail.

A.4 Statistics of the datasets used in the experiments

Table 9 shows the statistics of the datasets used in the experiments. For ACE2005 (ED), we refer to

	random seed	max_seq_len	batch size	epoch	dropout rate	learning rate	
						encoder layer	task layer
ACE 2005 (ED)	1	256	32	30	0.1	1e-5	2e-4
ACE 2005 (NER)	42	128	32	20	0.1	3e-5	6e-5
ACE 2004	42	128	32	30	0.1	3e-5	3e-4
OntoNotes 4.0	42	128	32	5	0.1	8e-6	8e-5
MSRA	42	128	32	20	0.1	3e-5	6e-5

Table 8: Hyper-parameter settings for each experiment.

		ACE 2005(ED)			ACE2005(NER)			ACE2004			OntoNote 4.0(Chinese)			MSRA		
		train	dev	test	train	dev	test	train	dev	test	train	dev	test	train	dev	test
sentences		13919	880	810	7294	971	1057	6200	745	812	15650	4301	4346	41729	4637	4366
spans	# total	4496	279	574	24703	3218	3029	22201	2514	3035	13367	6950	7684	70446	4157	6181
	# nested	-	-	-	5052 (20.45%)	598 (18.58%)	638 (21.06%)	5416 (24.40%)	623 (24.78%)	779 (25.67%)	-	-	-	-	-	-

Table 9: Statistics of the datasets used in the experiments. Spans are considered nested only if they are overlapped or nested in the different category.

the previous work[9] to process raw data, which follows standard data splitting strategy. NER datasets we used are provided in the previous SOTA work[10].

[9] https://github.com/thunlp/HMEAE

[10] https://github.com/ShannonAI/mrc-for-flat-nested-ner