# Packed Levitated Marker for Entity and Relation Extraction

Anonymous ACL submission

## Abstract

Recent entity and relation extraction works focus on investigating how to obtain a better span representation from the pre-trained encoder. However, a major limitation of existing works is that they ignore the interrelation between spans (pairs). In this work, we propose a novel span representation approach, named Packed Levitated Markers (PL-Marker), to consider the interrelation between the spans (pairs) by strategically packing the markers in the encoder. In particular, we propose a neighborhood-oriented packing strategy, which considers the neighbor spans integrally to better model the entity boundary information. Furthermore, for those more complicated span pair classification tasks, we design a subject-oriented packing strategy, which packs each subject and all its objects to model the interrelation between the same-subject span pairs. The experimental results show that, with the enhanced marker feature, our model advances baselines on six NER benchmarks, and obtains a 3.5%-3.6% strict relation F1 improvement with higher speed over previous state-of-the-art models on ACE04 and ACE05. All the code and data of this paper will be made publicly available.

## 1 Introduction

Recently, pre-trained language models (PLMs) (Devlin et al., 2019; Liu et al., 2019) have achieved significant improvements in Named Entity Recognition (NER, Luo et al. (2020); Fu et al. (2021)) and Relation Extraction (RE, Wadden et al. (2019); Zhou and Chen (2021)), two key sub-tasks of information extraction. Recent works (Wang et al., 2021c; Zhong and Chen, 2021) regard these two tasks as span classification or span pair classification, and thus focus on extracting better span representations from the PLMs.

Three span representation extraction methods are widely used: (1) **T-Concat** (Lee et al., 2017; Jiang et al., 2020) concatenates the representation of the span's boundary (start and end) tokens to obtain the span representation. It collects information at the token level but ignores the connection between boundary tokens of a span when they pass through the network; (2) **Solid Marker** (Soares et al., 2019; Zhou and Chen, 2021) explicitly insert two solid markers before and after the span to highlight the span in the input text. And it inserts two pair of markers to locate the subject and object of a span pair. However, the method cannot handle multiple span pairs at the same time because of its weakness in specifying the solid markers of a span pair from more than two pairs of markers in the sequence. (3) **Levitated Marker** (Zhong and Chen, 2021) first sets a pair of levitated markers to share the same position with the span's boundary tokens and then

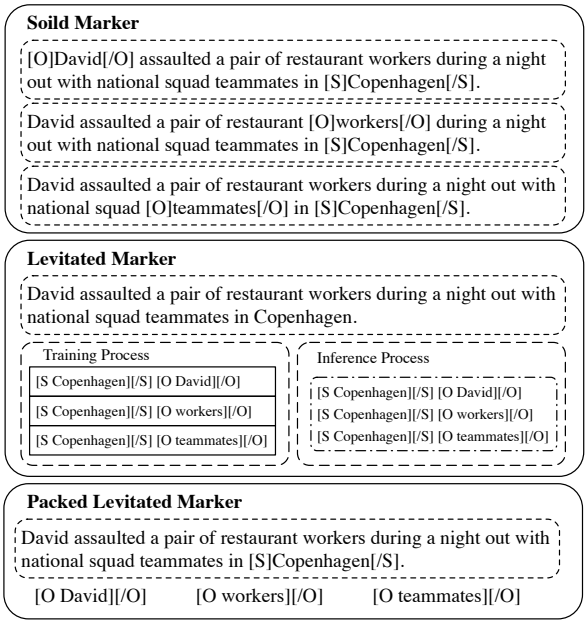


Figure 1: An example in the RE task. Solid Marker separately processes three pairs of spans with different insertions of markers. Levitated Marker processes the span pairs independently during training and processes them in batches during inference. Our proposed Packed Levitated Marker packs three objects for the same subject into an instance to process.

ties a pair of markers by a directional attention. To be specific, the markers within a pair are set to be visible to each other in the attention mask matrix, but not to the text token and other pairs of markers. Existing work (Zhong and Chen, 2021) simply replaces solid markers with levitated markers for an efficient batch computation, but sacrifices the model performance.

As the RE example shown in Figure 1, to correctly identify that David, workers and teammates are *located_in* Copenhagen, it is important to separate out that David *attacked* the restaurant workers and he had *social* relation with his teammates. However, prior works with markers (Zhong and Chen, 2021) independently processes the span pairs with different insertions of markers in the training phrase, and thus ignore interrelation between spans (pairs) (Sorokin and Gurevych, 2017; Luan et al., 2019; Wadden et al., 2019).

In this work, we introduce Packed Levitated Marker (PL-Marker), to model the interrelation between spans (pairs) by strategically packing levitated markers in the encoding phase. A key challenge of packing levitated markers together for span classification tasks is that the increasing number of inserted levitated markers would exacerbate the complexity of PLMs quadratically (Ye et al., 2021). Thus, we have to divide spans into several groups to control the length of each input sequence for a higher speed and feasibility. In this case, it is necessary to consider the neighbor spans integrally, which could help the model compare neighbor spans, *e.g.* the span with the same start token, to acquire a more precise entity boundary. Hence, we propose a neighborhood-oriented packing strategy, which packs the spans with the same start token into a training instance as much as possible to better distinguish the entity boundary.

For the more complicated span pair classification tasks, an ideal packing scheme is to pack all the span pairs together with multiple pairs of levitated markers, to model all the span pairs integrally. However, since each pair of levitated markers is already tied by directional attention, if we continue to apply directional attention to bind two pairs of markers, the levitated marker will not be able to identify its partner marker of the same span. Hence, we adopt a fusion of solid markers and levitated markers, and use a subject-oriented packing strategy to model the subject with all its related objects integrally. To be specific, we emphasize the subject span with solid markers and pack all its candidate object spans with levitated markers. Moreover, we apply an object-oriented packing strategy for an intact bidirectional modeling (Wu et al., 2020).

We examine the effect of PL-Marker on two typical span (pair) classification tasks, NER and end-to-end RE. The experimental results indicate that PL-Marker with neighborhood-oriented packing scheme performs much better than the model with random packing scheme on NER, which shows the necessity of considering the neighbor spans integrally. And our model also advances the T-Concat model on six NER benchmarks, which demonstrates the effectiveness of the feature obtained by span marker. Moreover, compared with the previous state-of-the-art RE model, our model gains a 3.5%-3.6% strict relation F1 improvement with higher speed on ACE04 and ACE05 and also achieves comparable performance on SciERC, which shows the importance of considering the interrelation between the subject-oriented span pairs.

## 2 Related Work

In recent years, span representation has attracted great attention from academia, which facilitates various NLP applications, such as named entity recognition (Ouchi et al., 2020), relation and event extraction (Luan et al., 2019), coreference resolution (Lee et al., 2017), semantic role labeling (He et al., 2018) and question answering (Lee et al., 2016). Existing methods to enhance span representation can be roughly grouped into three categories:

**Span Pre-training** The span pre-training approaches enhance the span representation for PLMs via span-level pre-training tasks. Sun et al. (2019); Lewis et al. (2020); Raffel et al. (2020) mask and learn to recover random contiguous spans rather than random tokens. Joshi et al. (2020) further learns to store the span information in its boundary tokens for downstream tasks.

**Knowledge Infusion** This series of methods focuses on infusing external knowledge into their models. Zhang et al. (2019); Peters et al. (2019); Wang et al. (2021a) learn to use the external entity embedding from the knowledge graph or the synonym net to acquire knowledge. Soares et al. (2019); Xiong et al. (2020); Wang et al. (2021b); Yamada et al. (2020) conduct specific entity-related pre-training to incorporate knowledge into their models with the help of Wikipeida anchor texts.

**Structural Extension** The structural extension methods add reasoning modules to the existing models, such as biaffine attention (Wang et al., 2021d), graph propagation (Wadden et al., 2019) and memory flow (Shen et al., 2021). With the support of modern pre-training encoders (*e.g.* BERT), the simple model with solid markers could achieve state-of-art results in RE (Zhou and Chen, 2021; Zhong and Chen, 2021). However, it is hard to specify the solid markers of a span pair from more than two pairs of markers in the sequence. Hence, previous work (Zhong and Chen, 2021) has to process span pairs independently, which is time-consuming and ignores the interrelation between the span pairs. In this work, we introduce the neighborhood-oriented and the subject-oriented packing strategies to take advantage of the levitated markers to provide an integral modeling on spans (pairs).

To our best knowledge, we are the first to apply the levitated markers on the NER. On the RE, the closest work to ours is the PURE (Approx.) (Zhong and Chen, 2021), which independently encodes each span pair with two pairs of levitated markers in the training phase and batches multiple pairs of markers to accelerate the inference process. Compared to their work, our model adopts a fusion subject-oriented packing scheme and thus handle multiple span pairs well in both the training and inference process. We detail the differences between our work and PURE in Section 4.4.2 and explain why our model performs better.

## 3 Method

In this section, we first introduce the architecture of the levitated marker. Then, we present how we pack the levitated marker to obtain the span representation and span pair representation.

### 3.1 Background: Levitated Marker

Levitated marker is used as an approximation of solid markers, which allows models to classify multiple pairs of entities simultaneously to accelerate the inference process (Zhong and Chen, 2021). A pair of levitated markers, associated with a span, consists of a start token marker and an end token marker. These two markers share the same position embedding with the start and end tokens of the corresponding span, while keeping the position id of original text tokens unchanged. In order to specify multiple pairs of levitated markers in parallel, a directional attention mask matrix is applied. Specifically, each levitated marker is visible to its partner marker within pair in the attention mask matrix, but not to the text tokens and other levitated markers. In the meantime, the levitated markers are able to attend to the text tokens to aggregate information for their associated spans.

### 3.2 Neighborhood-oriented Packing for Span

Benefiting from the parallelism of levitated markers, we can flexibly pack a series of related spans into a training instance. In practice, we append multiple associated levitated markers to an input sequence to conduct a comprehensive modeling on each span.

However, even though the entity length is restricted, some of the span classification tasks still contain a large number of candidate spans. Hence, we have to group the markers into several batches to equip the model with higher speed and feasibility in practice. To better model the connection between spans with the same start tokens, we adopt a neighborhood-oriented packing scheme. As shown in Figure 2, we first sort the pairs of levitated markers by taking the position of start marker as the first keyword and the position of end marker as the second keyword. After that, we split them into groups of size up to $K$ and thus gather adjacent spans into the same group. We packs each groups of markers and dispersedly process them in multiple runs.

Formally, given a sequence of $N$ text tokens, $X = \{x_1, \ldots, x_N\}$ and a maximum span length $L$, we define the candidate spans set as $S(X) = \{(1,1), .., (1,L), ..., (N, N-L), .., (N,N))\}$. We first divide $S(X)$ into multiple groups up to the size of $K$ in order. For example, we cluster $K$ spans, $\{(1,1),(1,2),...,(\lceil \frac{K}{L} \rceil, K - \lfloor \frac{K-1}{L} \rfloor * L)\}$, into a group $S_1$. We associate a pair of levitated markers to each span in $S_1$. Then, we provide the combined sequence of the text token and the inserted levitated markers to the PLM (*e.g.* BERT) to obtain the contextualized representations of the start token marker $H^{(s)} = \{h_i^{(s)}\}$ and that of the end token marker $H^{(e)} = \{h_i^{(e)}\}$. Here, $h_a^{(s)}$ and $h_b^{(e)}$ are associated with the span $s_i = (a, b)$, for which we obtain the span representations:

$$\psi(s_i) = [h_a^{(s)}; h_b^{(e)}] \quad (1)$$

where $[A; B]$ denotes the concatenation operation on the vector $A$ and $B$.

For instance, we apply the levitated marker to a typical overlapping span classification task, NER,

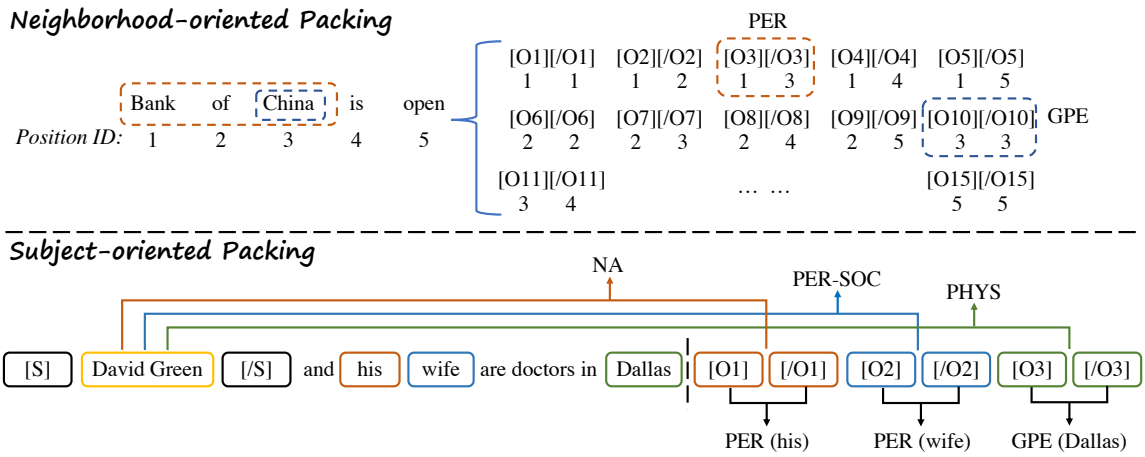


Figure 2: An overview of our neighborhood-oriented packing and subject-oriented packing strategies. [S][/S] are solid markers. [O][/O] are levitated markers. With a maximum group size, the neighborhood-oriented packing strategy clusters the neighbor spans, *e.g.* {(1,1),(1,2),...,(1,5)}, in the same group. The subject-oriented packing strategy encloses the subject span, *David Green*, with solid markers, applies levitated markers on its candidate object spans, *his*, *wife* and *Dallas*, and packs them into an instance.

which aims to assign an entity type or a non-entity type to each possible span in a sentence. We obtain the span representation from the PLM via the packed levitated markers and then combine the features of PL-Marker and T-Concat to better predict the entity type of the cadidate span.

### 3.3 Subject-oriented Packing for Span Pair

To obtain a span pair representation, a feasible method is to adopt levitated markers to emphasize a series of the subject and object spans simultaneously. Commonly, each pair of levitated markers is tied by the directional attention. But if we continue to apply directional attention to bind two pairs of markers, the levitated marker will not be able to identify its partner marker of the same span. Hence, as shown in Figure 2, our span pair model adopts a fusion subject-oriented packing scheme to offer an integral modeling for the same-subject spans.

Formally, given an input sequence $X$, a subject span, $s_i = (a, b)$ and its candidate object spans $(c_1, d_1), (c_2, d_2), ...(c_m, d_m)$, We insert a pair of solid markers [S] and [/S] before and after the subject span. Then, we apply levitated markers [O] and [/O] to all candidate object spans, and pack them into an instance. Let $\hat{X}$ denotes this modified sequence with inserted markers:

$$\hat{X} = ...[\text{S}], x_a, ..., x_b, [\text{/S}], ..., x_{c_1} \cup [\text{O1}], ...,$$
$$x_{d_1} \cup [\text{/O1}], ..., x_{c_2} \cup [\text{O2}], ..., x_{d_2} \cup [\text{/O2}]...,$$

where the tokens jointed by the symbol $\cup$ share the same position embedding. We apply a pre-trained encoder on $\hat{X}$ and finally obtain the span pair representation for $s_i = (a, b)$ and $s_j = (c, d)$:

$$\phi(s_i, s_j) = [h_{a-1}; h_{b+1}; h_c^{(s)}; h_d^{(e)}] \quad (2)$$

where $[\,;\,]$ denotes the concatenation operation. $h_{a-1}$ and $h_{b+1}$ denote the contextualized representation of the inserted solid markers for $s_i$; $h_c^{(s)}$ and $h_d^{(e)}$ are the contextualized representation of the inserted levitated markers for $s_j$.

Compared to the method that applies two pairs of solid markers on the subject and object respectively (Zhong and Chen, 2021), our fusion marker scheme replaces the solid markers with the levitated markers for the object span, which would impair the emphasis on the object span to some extent. To provide the supplemental information, we introduce an inverse relation from the object to the subject for a bidirectional prediction (Wu et al., 2020).

For instance, we evaluate our model on a typical span pair classification task, end-to-end RE, which concentrates on identifying whether all span pairs are related and their relation types. Following Zhong and Chen (2021), we first use a NER model to filter candidate entity spans, and then acquire the span pair representation of the filtered entity span pairs to predict the relation between them. Moreover, to build the connection between entity type and relation type, we add an auxiliary loss for predicting the type of object entity (Zhou and Chen, 2021; Han et al., 2021).

### 3.4 Complexity Analysis

Dominated by the large feed-forward network, the computation of PLM rises almost linearly with the increase in small sequence length (Dai et al., 2020; Ye et al., 2021). Gradually, as the sequence length continues to grow, the computation dilates quadratically due to the Self-Attention module (Vaswani et al., 2017). Obviously, the insertion of levitated markers extends the length of input sequence. For the span pair classification tasks, the number of candidate spans is relatively small , thus the increased computation is limited. For the span classification tasks, we group the markers into several batches, which can control the sequence length within the interval in which the complexity increases nearly linearly. For the NER, we enumerate candidate spans in a small-length sentence and then use its context words to expand the sentence to 512 tokens, for which the number of candidate spans in a sentence is usually less than the context length in practice. Hence, with a small number of packing groups, the complexity of PL-Marker is still near-linearly to the complexity of previous models.

Moreover, to further alleviate the inference cost, we adopt PL-Marker as a post-processing module of a two-stage model, in which it is used to identify entities from a small number of candidate entities proposed by a simpler and faster model.

## 4 Experiment

### 4.1 Experimental Setup

#### 4.1.1 Dataset

For the NER task, we conduct experiments on both flat and nested benchmarks. Firstly, on the flat NER, we adopt CoNLL03 (Sang and Meulder, 2003), OntoNotes 5.0 (Pradhan et al., 2013) and Few-NERD (Ding et al., 2021). Then, on the nested NER, we use ACE04 (Doddington et al., 2004), ACE05 (Walker et al., 2006) and SciERC (Luan et al., 2018). The three nested NER datasets are also used to evaluate the end-to-end RE. The details of the adopted datasets are shown in the Appendix.

#### 4.1.2 Evaluation Metrics

For NER task, we follow a span-level evaluation setting, where the entity boundary and entity type are required to correctly predicted. For the end-to-end RE, we report two evaluation metrics: (1) **Boundaries evaluation** (Rel) requires the model to correctly predict the boundaries of the subject entity and the object entity, and the entity relation; (2) **Strict evaluation** (Rel+) further requires the model to predict the entity types on the basis of the requirement of the boundary prediction. Moreover, following Wang et al. (2021d), we regard each symmetric relational instance as two directed relational instances.

#### 4.1.3 Implementation Details

We adopt *bert-base-uncased* (Devlin et al., 2019) and *albert-xxlarge-v1* (Lan et al., 2020) encoders for ACE04 and ACE05. For SciERC, we use the in-domain *scibert-scivocab-uncased* (Beltagy et al., 2019) encoder. For flat NER, we adopt *roberta-large* encoder. We also leverage the cross-sentence information (Luan et al., 2019; Luoma and Pyysalo, 2020), which extends each sentence by its context and ensures that the original sentence is located in the middle of the expanded sentence as much as possible. As discussed in Section 4.4.1, for the packing scheme on NER, we set the group size to 256 to improve efficiency. We run all experiments with 5 different seeds and report the average score. See the appendix for the standard deviations and the detailed training configuration.

Moreover, inspired by the success of prompt tuning (Brown et al., 2020; Schick and Schütze, 2021), we use the embedding of meaningful words instead of randomness to intialize the embedding of markers , denoted as **prompt init**. For NER, we initialize a pair of markers for the span with the words `[MASK]` and *entity*. For RE, we initialize a pair of solid markers for a subject with the words `[MASK]` and *subject* and initialize the markers for an object with the words `[MASK]` and *object*.

### 4.2 Named Entity Recognition

#### 4.2.1 Baselines

Our packing scheme allows the model to apply the levitated markers to process massive span pairs and to our best knowledge, we are the first to apply the levitated markers on the NER task. We compare our neighborhood-oriented packing scheme with the **Random Packing**, which randomly packs the candidate spans into groups. We adopt two common NER models: (1) **SeqTagger** (Devlin et al., 2019) regards NER as a sequence tagging task and applies a token-level classifier to distinguish the IOB2 tags for each word (Sang and Veenstra, 1999). (2) **T-Concat** (Jiang et al., 2020; Zhong and Chen, 2021) assigns an entity type or a non-entity type to each span based on its T-Concat span representation. Note that solid markers cannot deal with

| Model | CoNLL03 | OntoN5 | F-NERD |
|---|---|---|---|
| Ma and Hovy (2016) | 91.0 | 86.3 | - |
| Devlin et al. (2019) | 92.8 | 89.2 | 68.9 |
| Li et al. (2020) | 93.0 | 91.1 | - |
| Yu et al. (2020) | 93.5 | 91.3 | - |
| Yan et al. (2021) | 93.2 | 90.4 | - |
| SeqTagger (Our impl.) | 93.6 | 91.2 | 69.0 |
| T-Concat (Our impl.) | 93.0 | 91.7 | 70.6 |
| Random Packing | 93.9 | 91.7 | 61.5 |
| PL-Marker (Our model) | **94.0** | **91.9** | **70.9** |

Table 1: Micro F1 on the test set for the flat NER. OntoN5: OntoNotes 5.0; F-NERD: Few-NERD.

the overlapping spans simultaneously, thus it is too inefficient to apply solid markers independently on the NER task.

#### 4.2.2 Results

We show the flat NER results in the Table 1 and the nested NER results in the Ent column of Table 2, where PURE (Zhong and Chen, 2021) applies the T-Concat feature on its NER module. As follow, some observations are summarized from the experimental results: (1) The model with our neighborhood-oriented packing strategy outperforms the model with random packing strategy on all three flat NER datsets, especially obtaining a 9.4% improvement on Few-NERD. Few-NERD includes 325 candidate spans on average, while CoNLL03 and OntoNotes 5.0 only contain 90 and 174 respectively. It shows that the neighborhood-oriented packing strategy can well handle the dataset with more groups of markers to better model the interrelation among neighbor spans. (2) With the same pre-trained encoder, PL-Marker achieves an absolute F1 improvement of +0.1%-1.1% over T-Concat on all six NER benchmarks, which shows the advantage of levitated markers in aggregating span-wise representation for the entity type prediction; (3) PL-Marker outperforms SeqTagger by an absolute F1 of +0.4%, +0.7%, +1.9% in CoNLL03, OntoNote 5.0 and Few-NERD respectively, where CoNLL03, OntoNote 5.0 and Few-NERD contain 4, 18 and 66 entity types respectively. Such improvements prove the effectiveness of PL-Marker in handling diverse interrelation between entities of diverse types.

### 4.3 Relation Extraction

#### 4.3.1 Baselines

For the end-to-end RE, we compare our model, PL-Marker, with a series of state-of-the-art models. Here, we introduce two of the most representative works with T-Concat and Solid Markers span representation: (1) **DyGIE++** (Wadden et al., 2019) first acquires the T-Concat span representation, and then iteratively propagates coreference and relation type confidences through a span graph to refine the representation; (2) **PURE** (Zhong and Chen, 2021) adopts independent NER and RE models, where the RE model processes each possible entity pair in one pass. In their work, PURE (Full) adopts two pairs of solid markers to emphasize a span pair and the PURE (Approx) employs two pairs of levitated markers to underline the span pair.

#### 4.3.2 Results

As shown in Table 2, with the same $\text{BERT}_{\texttt{BASE}}$ encoder, our approach outperforms previous methods by strict F1 of +1.5% on ACE05 and +2.1% on ACE04. With the SciBERT encoder, our approach also achieves almost the best performance on SciERC. Using a larger encoder, $\text{ALBERT}_{\texttt{XXLARGE}}$, both of our NER and RE models are further improved. Compared to the previous state-of-the-art model, PURE (Full) (Zhong and Chen, 2021), our model gains a substantially +3.5% and +3.6% strict relation F1 improvement on ACE05 and ACE04 respectively. Such improvements over PURE indicate the effectiveness of modeling the interrelation between the same-subject or the same-object entity pairs in the training process.

### 4.4 Inference Speed

In this section, we compare the models' inference speed on an A100 GPU with a batch size of 32. We use the `BASE` size encoder for ACE05 and SciERC in the experiments and the `LARGE` size encoder for flat NER models.

#### 4.4.1 Speed of Span Model

We evaluate the inference speed of PL-Marker with different group size $K$ on CoNLL03 and Few-NERD. We also evaluate a cascade **Two-stage** model, which uses a fast `BASE`-size T-Concat model to filter candidate spans for our model. As shown in Table 3, PL-Marker achieves a 0.4 F1 improvement on CoNLL03 but sacrifices 60% speed compared to the SeqTagger model. And we observe that our proposed Two-stage model achieves similar performance to PL-Marker with 3.1x speedup on Few-NERD, which shows it is more efficient to use PL-Marker as a post-processing module to elaborate the coarse prediction from a simple model.

| Model | Encoder | Rep Type | ACE05 | | | ACE04 | | | SciERC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Ent | Rel | Rel+ | Ent | Rel | Rel+ | Ent | Rel | Rel+ |
| Li and Ji (2014) | - | - | 80.8 | 52.1 | 49.5 | 79.7 | 48.3 | 45.3 | - | - | - |
| SPtree (Miwa and Bansal, 2016) | LSTM | T | 83.4 | - | 55.6 | 81.8 | - | 48.4 | - | - | - |
| DYGIE (Luan et al., 2019)$^\diamond$ | ELMo | T | 88.4 | 63.2 | - | 87.4 | 59.7 | - | 65.2 | 41.6 | - |
| Multi-turn QA (Li et al., 2019) | $\text{BERT}_{\text{L}}$ | - | 84.8 | - | 60.2 | 83.6 | - | 49.4 | - | - | - |
| OneIE (Lin et al., 2020) | $\text{BERT}_{\text{L}}$ | T | 88.8 | 67.5 | - | - | - | - | - | - | - |
| DYGIE++ (Wadden et al., 2019)$^\diamond$ | $\text{BERT}_{\text{B}}$/ SciBERT | T | 88.6 | 63.4 | - | - | - | - | - | - | - |
| TriMF (Shen et al., 2021)$^\diamond$ | | T | 87.6 | 66.5 | 62.8 | - | - | - | **70.2** | **52.4** | - |
| UniRE (Wang et al., 2021d)$^\diamond$ | | T | 88.8 | - | 64.3 | 87.7 | - | 60.0 | 68.4 | - | 36.9 |
| PURE-F (Zhong and Chen, 2021)$^\diamond$ | | S | **90.1** | 67.7 | 64.8 | **89.2** | 63.9 | 60.1 | 68.9 | 50.1 | 36.8 |
| PURE-A (Zhong and Chen, 2021)$^\diamond$ | | L | - | 66.5 | - | - | - | - | - | 48.1 | - |
| PL-Marker (Our Model)$^\diamond$ | | S&L | 89.7 | **68.8** | **66.3** | 88.9 | **65.7** | **62.2** | 69.9 | 52.0 | **40.6** |
| TableSeq (Wang and Lu, 2020) | $\text{ALB}_{\text{XXL}}$ | T | 89.5 | 67.6 | 64.3 | 88.6 | 63.3 | 59.6 | - | - | - |
| UniRE (Wang et al., 2021d)$^\diamond$ | | T | 90.2 | - | 66.0 | 89.5 | - | 63.0 | - | - | - |
| PURE-F (Zhong and Chen, 2021)$^\diamond$ | | S | 90.9 | 69.4 | 67.0 | 90.3 | 66.1 | 62.2 | - | - | - |
| PL-Marker (Our Model)$^\diamond$ | | S&L | **91.1** | **72.3** | **70.5** | **90.4** | **69.2** | **65.8** | - | - | - |

Table 2: Overall entity and relation F1 scores on the test sets of ACE04, ACE05 and SciERC. The encoders used in different models: $\text{BERT}_{\text{B}}$ =$\text{BERT}_{\text{BASE}}$, $\text{BERT}_{\text{L}}$ = $\text{BERT}_{\text{LARGE}}$ = $\text{ALB}_{\text{XXL}}$ = $\text{ALBERT}_{\text{XXLARGE}}$. Specially, TriMF, UniRE, PURE and PL-Marker apply $\text{BERT}_{\text{BASE}}$ on ACE04/05 and apply the SciBERT on SciERC. $^\diamond$ denotes that the model leverages the cross-sentence information. Representation Type: T–*T-Concat*; S–*Solid Marker*; *L–Levitated Marker*. Model name abbreviation: PURE-F: PURE (Full); PURE-A: PURE (Approx.).

| Model | K | CoNLL03 | | Few-NERD | |
|---|---|---|---|---|---|
| | | Ent (F1) | Speed (sent/s) | Ent (F1) | Speed (sent/s) |
| SeqTagger | - | **93.6** | **138.7** | 69.0 | **142.0** |
| T-Concat | - | 93.0 | 137.2 | **70.6** | 126.8 |
| PL-Marker | 128 | **94.0** | **54.8** | **70.9** | 23.8 |
| | 256 | - | 39.6 | - | **25.8** |
| | 512 | - | 22.9 | - | 18.3 |
| Two-stage | 16 | 93.7 | **87.1** | 70.8 | **80.6** |
| | 32 | **94.0** | 83.3 | **70.9** | 79.8 |

Table 3: Micro F1 and efficiency on NER benchmarks with respect to the model and different packing group size $K$. We adopt a maximum span length of 8 for CoNLL03 and 16 for Few-NERD.

| Model | ACE05 | | SciERC | |
|---|---|---|---|---|
| | Rel (F1) | Speed (sent/s) | Rel (F1) | Speed (sent/s) |
| PURE (Full) | 67.7 | 76.5 | 50.1 | 88.3 |
| PURE (Approx.) | 66.5 | **593.7** | 48.8 | **424.2** |
| PL-Marker | **69.1** | 211.7 | **51.6** | 190.9 |

Table 4: Comparison of our RE model and PURE in relation F1 (boundaries) and speed. We report the result with BASE encoders. All models adopt the same entity input from the entity model of PURE.

In addition, when the group size grows to 512, PL-Marker slows down due to the increased complexity of the Transformer. Hence, we choose a group size of 256 in practice.

#### 4.4.2 Speed of Span Pair Model

We apply the subject-oriented and the object-oriented packing strategies on levitated markers for RE. Here, we compare our model with the other two marker-based models. Firstly, **PURE (Full)** (Zhong and Chen, 2021) applies solid markers to process each entity pair independently. Secondly, **PURE (Approx.)** packs the levitated markers of all entity pairs into an instance for batch computation. Since the performance and the running time of the above methods rely on the quality and the number of predicted entities, for a fair comparison, we adopt the same entity input from the entity model of PURE on all the RE models. Table 4 shows the relation F1 scores and the inference speed of the above three methods. On both datasets, our RE model, PL-Marker, achieves the best performance and PURE (Approx.) has highest efficiency in the inference process. Compared to the PURE (Full), our model obtains a 2.2x-2.8x speedup and better performance on ACE05 and SciERC. Compared to PURE (Approx.), our model achieves a 2.6%-2.8% relation F1 (boundaries) improvement on ACE05 and SciERC, which again demonstrates the effectiveness of our fusion markers and packing strategy. Overall, our model, with a novel subject-oriented packing strategy for markers, has been proven effective in practice, with satisfactory accuracy and affordable cost.

| **Named Entity Recognition** |
|---|
| **Text**: This is the Cross Strait program on CCTV International Channel. ... Candidates for the giant pandas to be presented to Taiwan as gifts from the mainland may increase. ...<br>**T-Concat**: (Cross Strait, WORK OF ART), (CCTV International Channel, ORG), (Taiwan, GPE)<br>**Our**: (Cross Strait, ORG), (CCTV International Channel, ORG), (Taiwan, GPE) |
| **Relation Extraction** |
| **Text**: *Liana* drove 10 hours from *Pennsylvania* to attend the rally in *Manhattan* with *her parents*<br>**PURE**: (Liana, located in, Manhattan)<br>**Our**: (Liana, located in, Manhattan), (her parents, located in, Manhattan) |

Table 5: Case study of our NER and RE model.

| Model | ACE05 | | SciERC | |
|---|---|---|---|---|
| | gold | e2e | gold | e2e |
| PL-Marker | **73.4** | **68.8** | **70.5** | **52.0** |
| w/o solid marker | 69.3 | 64.5 | 61.8 | 45.0 |
| w/o inverse relation | 70.5 | 65.2 | 63.7 | 46.3 |
| w/o entity type loss | 72.8 | 67.8 | 70.1 | 51.8 |
| w/o prompt init | 73.0 | 67.9 | 69.7 | 51.2 |

Table 6: The relation F1 (boundaries) on the test set of ACE05 and SciERC with different input features for the ablation study. gold: the gold entities; e2e: the entities are predicted by our entity model.

### 4.5 Case Study

We show several cases to compare our span model with T-Concat and to compare our span pair model with PURE (Full). As shown in Table 5, our span model could collect contextual information, such as *Taiwan* and *mainland*, for underlined span, *Cross Strait*, assisting in predicting its type as organization rather than work of art. Our span model learns to integrally consider the interrelation between the same-object relational facts in training phase, so as to successfully obtain the fact that both *Liana* and *her parents* are located in *Manhattan*.

### 4.6 Ablation Study

In this section, we conduct ablation studies to investigate the contribution of different components to our RE model, where we apply `BASE` size encoder in the experiments.

**Two pairs of Levitated Markers** We evaluate the *w/o solid marker* baseline, which applies two pairs of levitated markers on the subject and object respectively and packs all the span pairs into an instance. As shown in Table 6, compared to PL-Marker, the model without solid markers drops a huge 4.3%-8.7% F1 on ACE05 and SciERC when the golden entities are given. The result demonstrates that it is sub-optimal to continue to apply directional attention to bind two pairs of levitated markers, since a pair of levitated marker is already tied by the directional attention.

**Inverse Relation** We establish an inverse relation for each asymmetric relation for a bidirectional prediction. We evaluate the model without inverse relation, which replaces the constructed inverse relation with a non-relation type and adopts a unidirectional prediction. As shown in Table 6, the model without inverse relation significantly drops 2.9%-6.8% F1 on both datasets with the gold entities given, indicating the significance of modeling the information from the object entity to the subject entity in our asymmetric framework.

**Entity Type** We add an auxiliary entity type loss to RE model to introduce the entity type information. As shown in Table 6, when the gold entities are given, the model without entity type loss drops 0.6% F1 on ACE05 and 0.4% F1 on SciERC, which shows the importance of entity type information to RE. We also attempt to apply the entity type prediction from the RE model to refine the entity type prediction from the NER model. Please refer to the Appendix for details.

**Prompt Init** As illustrated in the implementation details, we initialize the embeddings of markers with the embeddings of practical words. Table 6 shows the results of the model without prompt initialization, which decreases 0.4%-0.8% F1 on ACE05 and SciERC when gold entities are given. It suggests that initializing markers with meaningful words helps the model to find the optimized direction during the training phase. As shown in the Appendix, the prompt initialization also brings a certain positive effects to our NER models.

## 5 Conclusion

In this work, we present a novel packed levitated markers, with a neighborhood-oriented packing strategy and a subject-oriented packing strategy, to obtain the span (pair) representation. Considering the interrelation between spans and span pairs, our model achieves the state-of-the-art F1 scores and a promising efficiency on both NER and RE tasks across six standard benchmarks. In future, we will further investigate how to generalize the marker-based span representation to more NLP tasks.

## References


Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *EMNLP-IJCNLP 2019*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS 2020*.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *NIPS 2020*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019*.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-nerd: A few-shot named entity recognition dataset. In *ACL/IJCNLP 2021*.

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie M. Strassel, and Ralph M. Weischedel. 2004. The automatic content extraction (ACE) program - tasks, data, and evaluation. In *LREC 2004*.

Jinlan Fu, Xuanjing Huang, and Pengfei Liu. 2021. Spanner: Named entity re-/recognition as span prediction. In *ACL/IJCNLP 2021*. Association for Computational Linguistics.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. PTR: prompt tuning with rules for text classification. *CoRR*, abs/2105.11259.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *ACL 2018*.

Zhengbao Jiang, Wei Xu, Jun Araki, and Graham Neubig. 2020. Generalizing natural language analysis through span-relation representations. In *ACL 2020*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8.

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR 2020*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP 2017*.

Kenton Lee, Tom Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *CoRR*, abs/1611.01436.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL 2020*. Association for Computational Linguistics.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *ACL 2014*.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *ACL 2020*.

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 *ACL 2019*.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *ACL 2020*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *EMNLP 2018*.

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 *NAACL-HLT 2019*.

Ying Luo, Fengshun Xiao, and Hai Zhao. 2020. Hierarchical contextualized representation for named entity recognition. In *AAAI 2020*.

Jouni Luoma and Sampo Pyysalo. 2020. Exploring cross-sentence contexts for named entity recognition with BERT. In *COLING 2020*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL 2016*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL 2016*.

Hiroki Ouchi, Jun Suzuki, Sosuke Kobayashi, Sho Yokoi, Tatsuki Kuribayashi, Ryuto Konno, and Kentaro Inui. 2020. Instance-based learning of span representations: A case study through named entity recognition. In *ACL 2020*.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *EMNLP-IJCNLP 2019*. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *CoNLL 2013*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *HLT-NAACL 2003*.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *EACL 1999*.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL 2021*. Association for Computational Linguistics.

Yongliang Shen, Xinyin Ma, Yechun Tang, and Weiming Lu. 2021. A trigger-sense memory flow framework for joint entity and relation extraction. In *WWW 2021*.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *ACL 2019*.

Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *EMNLP 2017*.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. ERNIE: enhanced representation through knowledge integration. *CoRR*, abs/1904.09223.

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 2017*.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *EMNLP-IJCNLP 2019*.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. In *Linguistic Data Consortium*.

Jue Wang and Wei Lu. 2020. Two are better than one: Joint entity and relation extraction with table-sequence encoders. In *EMNLP 2020*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021a. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of ACL/IJCNLP 2021*, Findings of ACL. Association for Computational Linguistics.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *TACL 2021*.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021c. Automated concatenation of embeddings for structured prediction. In *ACL/IJCNLP 2021*.

Yijun Wang, Changzhi Sun, Yuanbin Wu, Hao Zhou, Lei Li, and Junchi Yan. 2021d. Unire: A unified label space for entity relation extraction. In *ACL/IJCNLP 2021*.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. Corefqa: Coreference resolution as query-based span prediction. In *ACL 2020*.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2020. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *ICLR 2020*.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *ACL/IJCNLP 2021*. Association for Computational Linguistics.

Deming Ye, Yankai Lin, Yufei Huang, and Maosong Sun. 2021. TR-BERT: dynamic token reduction for accelerating BERT inference. In *NAACL-HLT 2021*.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *ACL 2020*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *ACL 2019*.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *NAACL-HLT 2021*.

Wenxuan Zhou and Muhao Chen. 2021. An improved baseline for sentence-level relation extraction. *CoRR*, abs/2102.01373.

## A Training Configuration

We follow Luan et al. (2019) to split ACE04 into 5 folds and split ACE05 into train, development, and test sets. For other datasets, we adopt the official split. Table 7 shows the statistics of each dataset. For the bidirectional prediction on RE, we set the forward and inverse relation of symmetric labels to be consistent. The symmetric labels include the *PER-SOC* in the ACE04/05 and the *Compare* and *Conjunction* in the SciERC.

We train all the models with Adam optimizer (Kingma and Ba, 2015) and 10% warming-up steps. And we adopt a learning rate of 2e-5 for models with `BASE` size and a learning rate of 1e-5 for models with `LARGE` or `XXLARGE` size. We run all experiments with 5 different seeds (42, 43, 44, 45, 46).

For NER model, we set the maximum length of expanded sentence $C$ as 512. For RE model, we set $C$ as 256 for ACE05 and SciERC and set $C$ as 384 for ACE04, whose original sentence might exceed 256 tokens. To enumerate possible spans, we set the maximum span length $L$ as 16 for OntoNote 5.0 and Few-NERD and set 8 for the other datasets.

For the NER of CoNLL03 and the RE of ACE05, we search the batch size in [4,8,16,32] and observe the model with a batch size of 8 achieves a sightly better performance. Hence, we choose a batch size of 8 for all the datasets. We search the number of epochs in [3,5,8,10,15,50] for all the datasets and finally choose 8 for CoNLL03, 5 for OntoNote 5.0, 3 for Few-NERD, 10 for ACE05-NER, 15 for ACE04-NER, 50 for SciERC-NER and 10 for all the RE models.

## B NER Results

We illustrate the span representation for NER in Figure 3. We show the average scores of the baselines and PL-Marker on flat NER with standard deviations in Table 8.

| Dataset | #Sents | #Ents (#Types) | #Rels (#Types) |
|---|---|---|---|
| CoNLL03 | 22.1k | 35.1k (4) | - |
| OntoNotes 5.0 | 103.8k | 161.8k (18) | - |
| Few-NERD | 188.2k | 491.7k (66) | - |
| ACE05 | 14.5k | 38.3k (7) | 7.1k (6) |
| ACE04 | 8.7k | 22.7k (7) | 4.1k (6) |
| SciERC | 2.7k | 8.1k (6) | 4.6k (7) |

Table 7: The statistics of the adopted datasets.

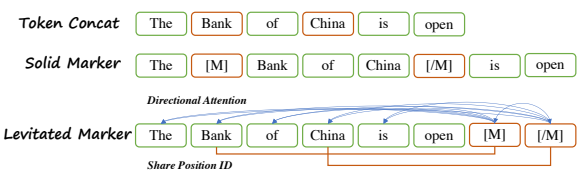


Figure 3: An illustration of T-Concat, Solid Marker and Levitated Marker in the entity typing task. We highlight the attention direction for the levitated marker and omit the bidirectional attention line between other token pairs. Token Concat conveys the representation of edged token, [Bank] and [China], through the classifier, while Solid Marker and Levitated Marker employ the features of two markers, [M] and [/M] for classification.

## C RE Results

We show the average scores of PL-Marker on RE with standard deviations in Table 9.

### C.1 Refine Entity Type

As shown in Table 10, using the entity type predicted by the RE model brings +0.6% and -0.3% strict relation F1 on ACE05 and SciERC respectively. We observe that the most frequent entity type pair for each relation occupies 48.5% for ACE05, 52.0% for ACE04 and 19.1% for SciERC, which shows that the relation is more relevant to the entity type in ACE05 than that in SciERC. Hence, we only use the RE model to refine the NER results for ACE04 and ACE05.

## D Prompt Initialization for NER

As shown in Table 11, using meaningful words, *entity* and `[MASK]`, as prompt to initialize the markers can bring a slight improvement to all six NER benchmarks.

| Model | CoNLL03 | OntoNotes 5 | Few-NERD |
|---|---|---|---|
| SeqTagger | $93.6_{\pm 0.1}$ | $91.2_{\pm 0.2}$ | $69.0_{\pm 0.1}$ |
| Token Concat | $93.0_{\pm 0.2}$ | $91.7_{\pm 0.1}$ | $70.6_{\pm 0.1}$ |
| Random Packing | $93.9_{\pm 0.2}$ | $91.7_{\pm 0.2}$ | $61.5_{\pm 0.1}$ |
| PL-Marker | $\mathbf{94.0}_{\pm 0.1}$ | $\mathbf{91.9}_{\pm 0.1}$ | $\mathbf{70.9}_{\pm 0.1}$ |

Table 8: Overall entity and relation F1 scores of the baselines and PL-Marker on the test set of CoNLL03, OntoNotes 5.0 and Few-NERD. We report average scores across five random seeds, with standard deviations as subscripts.

| Dataset | Encoder | Ent | Rel | Rel+ |
|---|---|---|---|---|
| **ACE05** | $\text{BERT}_{\text{B}}$ | $89.7_{\pm 0.27}$ | $68.8_{\pm 0.50}$ | $66.3_{\pm 0.45}$ |
| | $\text{ALB}_{\text{XXL}}$ | $\mathbf{91.1}_{\pm 0.29}$ | $\mathbf{72.3}_{\pm 0.88}$ | $\mathbf{70.5}_{\pm 0.69}$ |
| **ACE04** | $\text{BERT}_{\text{B}}$ | $88.9_{\pm 0.70}$ | $65.7_{\pm 1.98}$ | $62.2_{\pm 1.67}$ |
| | $\text{ALB}_{\text{XXL}}$ | $\mathbf{91.1}_{\pm 0.45}$ | $\mathbf{72.3}_{\pm 2.02}$ | $\mathbf{70.5}_{\pm 1.58}$ |
| **SciERC** | SciBERT | $\mathbf{69.9}_{\pm 0.71}$ | $\mathbf{52.0}_{\pm 0.61}$ | $\mathbf{40.6}_{\pm 0.56}$ |

Table 9: Overall entity and relation F1 scores of PL-Marker on the test set of ACE04, ACE05 and SciERC. $\text{BERT}_{\text{B}}$ denotes $\text{BERT}_{\text{BASE}}$; ALB denotes $\text{ALBERT}_{\text{XXLARGE}}$; We report average scores across five random seeds with standard deviations as subscripts.

| Entity Type Source | ACE05 | | SciERC | |
|---|---|---|---|---|
| | Ent | Rel+ | Ent | Rel+ |
| NER Model | **89.8** | 65.7 | **69.9** | **40.6** |
| +RE Model Refine | 89.7 | **66.3** | 69.5 | 40.3 |

Table 10: The entity F1 and the strict relation F1 on the test set of ACE05 and SciERC when the RE model is used to refine the NER prediction or when it is not used.

| Init Strategy | CoNLL03 | OntoNote 5.0 | Few-NRED |
|---|---|---|---|
| Random | $93.9_{\pm 0.12}$ | $91.8_{\pm 0.12}$ | $70.8_{\pm 0.11}$ |
| Prompt | $\mathbf{94.0}_{\pm 0.09}$ | $\mathbf{91.9}_{\pm 0.08}$ | $\mathbf{70.9}_{\pm 0.05}$ |
| **Init Strategy** | **ACE05** | **ACE04** | **SciERC** |
| Random | $91.0_{\pm 0.28}$ | $90.3_{\pm 0.45}$ | $69.4_{\pm 0.50}$ |
| Prompt | $\mathbf{91.1}_{\pm 0.29}$ | $\mathbf{90.4}_{\pm 0.42}$ | $\mathbf{69.9}_{\pm 0.71}$ |

Table 11: The entity F1 on test set of NER datasets when the PL-Marker is initialized with prompt or when it is initialized randomly. We use the $\text{RoBERTa}_{\text{LARGE}}$ for flat NER datasets and $\text{ALBERT}_{\text{XXLARGE}}$ for the nested NER datasets.