A Partition Filter Network for Joint Entity and Relation Extraction

Zhiheng Yan[1], **Chong Zhang**[1], **Jinlan Fu**[1,2], **Qi Zhang**[1*] **and Zhongyu Wei**[3]

[1]School of Computer Science, Shanghai Key Laboratory of Intelligent Information Processing,
Fudan University, Shanghai, China
[2]National University of Singapore, Singapore
[3]School of Data Science, Fudan University, Shanghai, China
`{zhyan20, chongzhang20, qz, zywei}@fudan.edu.cn`
`jinlanjonna@gmail.com`

Abstract

In joint entity and relation extraction, existing work either sequentially encode task-specific features, leading to an imbalance in inter-task feature interaction where features extracted later have no direct contact with those that come first. Or they encode entity features and relation features in a parallel manner, meaning that feature representation learning for each task is largely independent of each other except for input sharing. We propose a partition filter network to model two-way interaction between tasks properly, where feature encoding is decomposed into two steps: partition and filter. In our encoder, we leverage two gates: entity and relation gate, to segment neurons into two task partitions and one shared partition. The shared partition represents inter-task information valuable to both tasks and is evenly shared across two tasks to ensure proper two-way interaction. The task partitions represent intra-task information and are formed through concerted efforts of both gates, making sure that encoding of task-specific features is dependent upon each other. Experiment results on six public datasets show that our model performs significantly better than previous approaches. In addition, contrary to what previous work has claimed, our auxiliary experiments suggest that relation prediction is contributory to named entity prediction in a non-negligible way. The source code can be found at `https://github.com/Coopercoppers/PFN`.

1 Introduction

Joint entity and relation extraction intend to simultaneously extract entity and relation facts in the given text to form relational triples as (s, r, o). The extracted information provides a supplement to many studies, such as knowledge graph construction (Riedel et al., 2013), question

---

[*] Corresponding author.

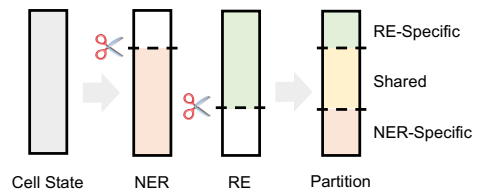answering (Diefenbach et al., 2018) and text summarization (Gupta and Lehal, 2010).

Figure 1: Partition process of cell neurons. Entity and relation gate are used to divide neurons into task-related and task-unrelated ones. Neurons relating to both tasks form the shared partition while the rest form two task partitions.

Conventionally, Named Entity Recognition (NER) and Relation Extraction (RE) are performed in a pipelined manner (Zelenko et al., 2002; Chan and Roth, 2011). These approaches are flawed in that they do not consider the intimate connection between NER and RE. Also, error propagation is another drawback of pipeline methods. In order to conquer these issues, joint extracting entity and relation is proposed and demonstrates stronger performance on both tasks. In early work, joint methods mainly rely on elaborate feature engineering to establish interaction between NER and RE (Yu and Lam, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014). Recently, end-to-end neural network has shown to be successful in extracting relational triples (Zeng et al., 2014; Gupta et al., 2016; Katiyar and Cardie, 2017; Shen et al., 2021) and has since become the mainstream of joint entity and relation extraction.

According to their differences in encoding task-specific features, most of the existing methods can be divided into two categories: sequential encoding and parallel encoding. In sequential encoding, task-specific features are generated sequentially, which means features extracted first are not affected by those that are extracted later. Zeng et al. (2018) and Wei et al. (2020) are

typical examples of this category. Their methods extract features for different tasks in a predefined order. In parallel encoding, task-specific features are generated independently using shared input. Compared with sequential encoding, models build on this scheme do not need to worry about the implication of encoding order. For example, Fu et al. (2019) encodes entity and relation information separately using common features derived from their GCN encoder. Since both task-specific features are extracted through isolated sub-modules, this approach falls into the category of parallel encoding.

However, both encoding designs above fail to model two-way interaction between NER and RE tasks properly. In sequential encoding, interaction is only unidirectional with a specified order, resulting in different amount of information exposed to NER and RE task. In parallel encoding, although encoding order is no longer a concern, interaction is only present in input sharing. Considering adding two-way interaction in feature encoding, we adopt an alternative encoding design: joint encoding. This design encodes task-specific features jointly with a single encoder where there should exist some mutual section for inter-task communication.

In this work, we instantiate joint encoding with a partition filter encoder. Our encoder first sorts and partitions each neuron according to its contribution to individual tasks with entity and relation gates. During this process, two task partitions and one shared partition are formed (see figure 1). Then individual task partitions and shared partition are combined to generate task-specific features, filtering out irrelevant information stored in the opposite task partition.

Task interaction in our encoder is achieved in two ways: First, the partitions, especially the task-specific ones, are formed through concerted efforts of entity and relation gates, allowing for interaction between the formation of entity and relation features determined by these partitions. Second, the shared partition, which represents information useful to both task, is equally accessible to the formation of both task-specific features, ensuring balanced two-way interaction. The contributions of our work are summarized below:

1. We propose partition filter network, a framework designed specifically for joint encoding. This method is capable of encoding task-specific features and guarantees proper two-way interaction between NER and RE.

2. We conduct extensive experiments on six datasets. The main results show that our method is superior to other baseline approaches, and the ablation study provides insight into what works best for our framework.

3. Contrary to what previous work has claimed, our auxiliary experiments suggest that relation prediction is contributory to named entity prediction in a non-negligible way.

2 Related Work

In recent years, joint entity and relation extraction approaches have been focusing on tackling triple overlapping problem and modelling task interaction. Solutions to these issues have been explored in recent works (Zheng et al., 2017; Zeng et al., 2018, 2019; Fu et al., 2019; Wei et al., 2020). The triple overlapping problem refers to triples sharing the same entity (SEO, i.e. SingleEntityOverlap) or entities (EPO, i.e. EntityPairOverlap). For example, In "Adam and Joe were born in the USA", since triples (Adam, birthplace, USA) and (Joe, birthplace, USA) share only one entity "USA", they should be categorized as SEO triples; or in "Adam was born in the USA and lived there ever since", triples (Adam, birthplace, USA) and (Adam, residence, USA) share both entities at the same time, thus should be categorized as EPO triples. Generally, there are two ways in tackling the problem. One is through generative methods like seq2seq (Zeng et al., 2018, 2019) where entity and relation mentions can be decoded multiple times in output sequence, another is by modeling each relation separately with sequences (Wei et al., 2020), graphs (Fu et al., 2019) or tables (Wang and Lu, 2020). Our method uses relation-specific tables (Miwa and Sasaki, 2014) to handle each relation separately.

Task interaction modeling, however, has not been well handled by most of the previous work. In some of the previous approaches, Task interaction is achieved with entity and relation prediction sharing the same features (Tran and Kavuluru, 2019; Wang et al., 2020b). This could be problematic as information about entity and relation could sometimes be contradictory. Also, as models that use sequential encoding (Bekoulis et al., 2018b; Eberts and Ulges, 2019; Wei et al., 2020) or parallel encoding (Fu et al., 2019) lack proper two-way interaction in feature extraction, predictions made on these features suffer the

problem of improper interaction. In our work, the partition filter encoder is built on joint encoding and is capable of handling communication of inter-task information more appropriately to avoid the problem of sequential and parallel encoding (exposure bias and insufficient interaction), while keeping intra-task information away from the opposite task to mitigate the problem of negative transfer between the tasks.

3 Problem Formulation

Our framework split up joint entity and relation extraction into two sub-tasks: NER and RE. Formally, Given an input sequence $s = \{w_1, \dots, w_L\}$ with L tokens, w_i denotes the i-th token in sequence s. For NER, we aim to extract all typed entities whose set is denoted as S, where $\langle w_i, e, w_j \rangle \in S$ signifies that token w_i and w_j are the start and end token of an entity typed $e \in \mathcal{E}$. $\mathcal{E}$ represents the set of entity types. Concerning RE, the goal is to identify all head-only triples whose set is denoted as T, each triple $\langle w_i, r, w_j \rangle \in T$ indicates that tokens w_i and w_j are the corresponding start token of subject and object entity with relation $r \in \mathcal{R}$. $\mathcal{R}$ represents the set of relation types. Combining the results from both NER and RE, we should be able to extract relational triples with complete entity spans.

4 Model

We describe our model design in this section. Our model consists of a partition filter encoder and two task units, namely NER unit and RE unit. The partition filter encoder is used to generate task-specific features, which will be sent to task units as input for entity and relation prediction. We will discuss each component in detail in the following three sub-sections.

4.1 Partition Filter Encoder

Similar to LSTM, the partition filter encoder is a recurrent feature encoder with information stored in intermediate memories. In each time step, the encoder first divides neurons into three partitions: entity partition, relation partition and shared partition. Then it generates task-specific features by selecting and combining these partitions, filtering out information irrelevant to each task. As shown in figure 2, this module is designed specifically to jointly extract task-specific features, which strictly follows two steps: partition and filter.

Partition This step performs neuron partition to divide cell neurons into three partitions: Two task partitions storing intra-task information, namely entity partition and relation partition, as well as one shared partition storing inter-task information. The neuron to be divided are candidate cell $\tilde{c}_t$ representing current information and previous cell c_{t-1} representing history information. c_{t-1} is the direct input from the last time step and $\tilde{c}_t$ is calculated in the same manner as LSTM:

$$\tilde{c}_t = \tanh(\text{Linear}([x_t; h_{t-1}])) \qquad (1)$$

where Linear stands for the operation of linear transformation.

We leverage entity gate $\tilde{e}$ and relation gate $\tilde{r}$, which are referred to as master gates in (Shen et al., 2019), for neuron partition. As illustrated in figure 1, each gate, which represents one specific task, will divide neurons into two segments according to their usefulness to the designated task. For example, entity gate $\tilde{e}$ will separate neurons into two partitions: NER-related and NER-unrelated. The shared partition is formed by combining partition results from both gates. Neurons in the shared partition can be regarded as information valuable to both tasks. In order to model two-way interaction properly, inter-task information in the shared partition is evenly accessible to both tasks (which will be discussed in the filter subsection). In addition, information valuable to only one task is invisible to the opposing task and will be stored in individual task partitions. The gates are calculated using cummax activation function $cummax(\cdot) = cumsum(softmax(\cdot))$[1], whose output can be seen as approximation of a binary gate with the form of $(0, \dots, 0, 1, \dots, 1)$:

$$\begin{aligned} \tilde{e} &= \text{cummax}(\text{Linear}([x_t; h_{t-1}])) \\ \tilde{r} &= 1 - \text{cummax}(\text{Linear}([x_t; h_{t-1}])) \end{aligned} \qquad (2)$$

The intuition behind equation (2) is to identify two cut-off points, displayed as scissors in figure 2, which naturally divide a set of neurons into three segments.

As a result, the gates will divide neurons into three partitions, entity partition ρ_e, relation partition ρ_r and shared partition ρ_s. Partitions for

---

[1] $cumsum(x_1, x_2, \dots, x_{n-1}, x_n) = (x_1, x_1 + x_2, \dots, x_1 + x_2 + \dots + x_{n-1}, x_1 + x_2 + \dots + x_{n-1} + x_n)$.

NER Units   RE Units

Global Feature

NER-specific Feature
Shared Feature
RE-specific Feature

Partition Filter | Partition Filter | Partition Filter | Partition Filter | Partition Filter

x_1 x_2 x_3 x_4 x_5

(a) Framework of Partition Filter Network
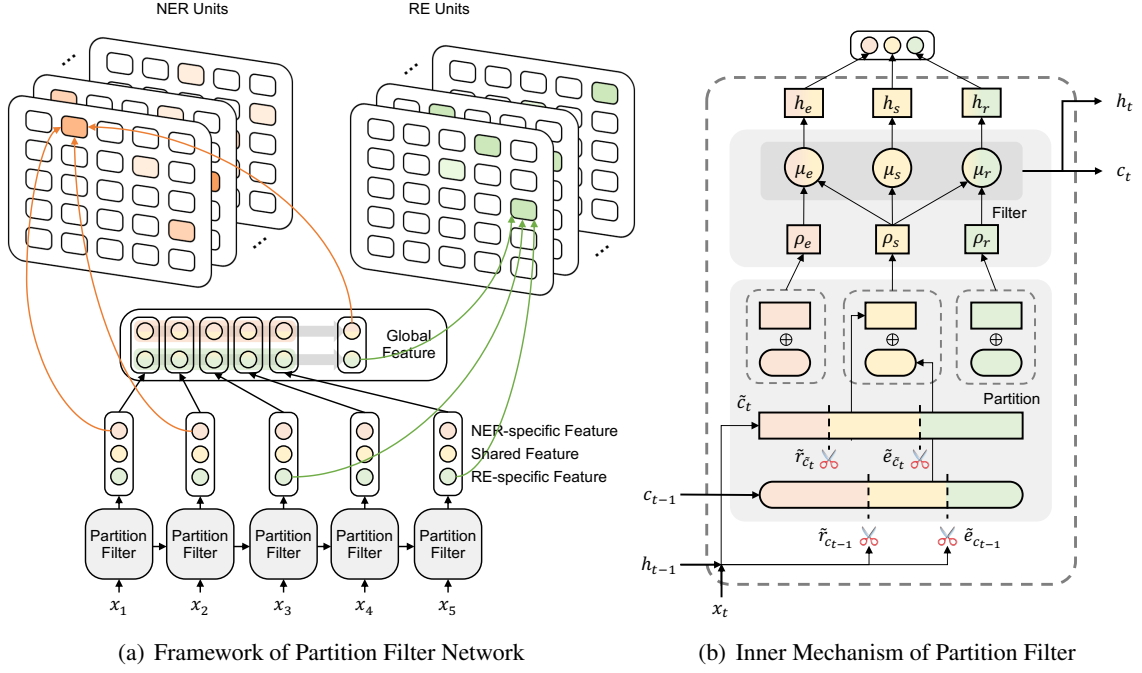
(b) Inner Mechanism of Partition Filter

Figure 2: (a) Overview of PFN. The framework consists of three components: partition filter encoder, NER unit and RE unit. In task units, we use table-filling for word pair prediction. Orange, yellow and green represents NER-related, shared and RE-related component or features. (b) Detailed depiction of partition filter encoder in one single time step. We decompose feature encoding into two steps: partition and filter (shown in the gray area). In partition, we first segment neurons into two task partitions and one shared partition. Then in filter, partitions are selected and combined to form task-specific features and shared features, filtering out information irrelevant to each task.

previous cell c_{t-1} are formulated as below:[2]

$$\rho_{s,c_{t-1}} = \tilde{e}_{c_{t-1}} \circ \tilde{r}_{c_{t-1}}$$
$$\rho_{e,c_{t-1}} = \tilde{e}_{c_{t-1}} - \rho_{s,c_{t-1}} \qquad (3)$$
$$\rho_{r,c_{t-1}} = \tilde{r}_{c_{t-1}} - \rho_{s,c_{t-1}}$$

Note that if you add up all three partitions, the result is not equal to one. This guarantees that in forward message passing, some information is discarded to ensure that message is not overloaded, which is similar to the forgetting mechanism in LSTM.

Then, we aggregate partition information from both target cells, and three partitions are formed as a result. For all three partitions, we add up all related information from both cells:

$$\rho_e = \rho_{e,c_{t-1}} \circ c_{t-1} + \rho_{e,\tilde{c}_t} \circ \tilde{c}_t$$
$$\rho_r = \rho_{r,c_{t-1}} \circ c_{t-1} + \rho_{r,\tilde{c}_t} \circ \tilde{c}_t \qquad (4)$$
$$\rho_s = \rho_{s,c_{t-1}} \circ c_{t-1} + \rho_{s,\tilde{c}_t} \circ \tilde{c}_t$$

Filter We propose three types of memory block: entity memory, relation memory and shared memory. Here we denote μ_e as entity memory, μ_r as

---

[2]The calculation for candidate cell $\tilde{c}_t$ is practically identical to equation (3) and therefore not shown.

relation memory and μ_s as shared memory. In μ_e, information in entity partition and shared partition are selected. In contrast, information in relation partition, which we assume is irrelevant or even harmful to named entity recognition task, is filtered out. The same logic applies to μ_r as well, where information in entity partition is filtered out and the rest is kept. In addition, information in shared partition will be stored in μ_s:

$$\mu_e = \rho_e + \rho_s; \ \mu_r = \rho_r + \rho_s; \ \mu_s = \rho_s \qquad (5)$$

Note that inter-task information in the shared partition is accessible to both entity memory and relation memory, allowing balanced interaction between NER and RE. Whereas in sequential and parallel encoding, relation features have no direct impact on the formation of entity features.

After updating information in each memory, entity features h_e, relation features h_r and shared features h_s are generated with corresponding memories:

$$h_e = \tanh(\mu_e)$$
$$h_r = \tanh(\mu_r) \qquad (6)$$
$$h_s = \tanh(\mu_s)$$

Following the partition and filter steps, information

in all three memories is used to form cell state c_t, which will then be used to generate hidden state h_t (The hidden and cell state at time step t are input to the next time step):

$$c_t = \text{Linear}([\mu_{e,t}; \mu_{r,t}; \mu_{s,t}]) \quad (7)$$
$$h_t = \tanh(c_t)$$

4.2 Global Representation

In our model, we employ a unidirectional encoder for feature encoding. The backward encoder in the bidirectional setting is replaced with task-specific global representation to capture the semantics of future context. Empirically this shows to be more effective. For each task, global representation is the combination of task-specific features and shared features computed by:

$$h_{ge,t} = \tanh(\text{Linear}[h_{e,t}; h_{s,t}])$$
$$h_{gr,t} = \tanh(\text{Linear}[h_{r,t}; h_{s,t}])$$
$$h_{ge} = \text{maxpool}(h_{ge,1}, \ldots, h_{ge,L}) \quad (8)$$
$$h_{gr} = \text{maxpool}(h_{gr,1}, \ldots, h_{gr,L})$$

4.3 Task Units

Our model consists of two task units: NER unit and RE unit. In NER unit, the objective is to identify and categorize all entity spans in a given sentence. More specifically, the task is treated as a type-specific table filling problem. Given a entity type set $\mathcal{E}$, for each type k, we fill out a table whose element e_{ij}^k represents probability of word w_i and word w_j being start and end position of an entity with type k. For each word pair (w_i, w_j), we concatenate word-level entity features h_i^e and h_j^e, as well as sentence-level global features h_{ge} before feeding it into a fully-connected layer with ELU activation to get entity span representation h_{ij}^e:

$$h_{ij}^e = \text{ELU}(\text{Linear}([h_i^e; h_j^e; h_{ge}])) \quad (9)$$

With the span representation, we can predict whether the span is an entity with type k by feeding it into a feed forward neural layer:

$$e_{ij}^k = p(e = \langle w_i, k, w_j \rangle | e \in S)$$
$$= \sigma(\text{Linear}(h_{ij}^e)), \forall k \in \mathcal{E} \quad (10)$$

where σ represents sigmoid activation function.

Computation in RE unit is mostly symmetrical to NER unit. Given a set of gold relation triples denoted as T, this unit aims to identify all triples in the sentence. We only predict starting word of each entity in this unit as entity span prediction is already covered in NER unit. Similar to NER, we consider relation extraction as a relation-specific table filling problem. Given a relation label set $\mathcal{R}$, for each relation $l \in \mathcal{R}$, we fill out a table whose element r_{ij}^l represents the probability of word w_i and word w_j being starting word of subject and object entity. In this way, we can extract all triples revolving around relation l with one relation table. For each triple (w_i, l, w_j), similar to NER unit, triple representation h_{ij}^r and relation score r_{ij}^l are calculated as follows:

$$h_{ij}^r = \text{ELU}(\text{Linear}([h_i^r; h_j^r; h_{gr}]))$$
$$r_{ij}^l = p(r = \langle w_i, l, w_j \rangle | r \in T) \quad (11)$$
$$= \sigma(\text{Linear}(h_{ij}^r)), \forall l \in \mathcal{R}$$

4.4 Training and Inference

For a given training dataset, the loss function L that guides the model during training consists of two parts: L_{ner} for NER unit and L_{re} for RE unit:

$$L_{ner} = \sum_{\hat{e}_{ij}^k \in S} \text{BCELoss}(e_{ij}^k, \hat{e}_{ij}^k)$$
$$L_{re} = \sum_{\hat{r}_{ij}^l \in T} \text{BCELoss}(r_{ij}^l, \hat{r}_{ij}^l) \quad (12)$$

$\hat{e}_{ij}^k$ and $\hat{r}_{ij}^l$ are respectively ground truth label of entity table and relation table. e_{ij}^k and r_{ij}^l are the predicted ones. We adopt BCELoss for each task[3]. The training objective is to minimize the loss function L, which is computed as $L_{ner} + L_{re}$.

During inference, we extract relational triples by combining results from both NER and RE unit. For each legitimate triple prediction $(s_{i,j}^k, l, o_{m,n}^{k'})$ where l is the relation label, k and k' are the entity type labels, and the indexes i, j and m, n are respectively starting and ending index of subject entity s and object entity o, the following conditions should be satisfied:

$$e_{ij}^k \geq \lambda_e; \ e_{mn}^{k'} \geq \lambda_e; \ r_{im}^l \geq \lambda_r \quad (13)$$

λ_e and λ_r are threshold hyper-parameters for entity and relation prediction, both set to be 0.5 without further fine-tuning.

5 Experiment

5.1 Dataset, Evaluation and Implementation Details

We evaluate our model on six datasets. NYT (Riedel et al., 2010), WebNLG (Zeng et al., 2018),

---

[3]$\text{BCELoss}(x, y) = -(y \log x + (1 - y) \log(1 - x))$.

ADE (Gurulingappa et al., 2012), SciERC (Luan et al., 2018), ACE04 and ACE05 (Walker et al., 2006). Descriptions of the datasets can be found in Appendix A.

Following previous work, we assess our model on NYT/WebNLG under **partial match**, where only the tail of an entity is annotated. Besides, as entity type information is not annotated in these datasets, we set the type of all entities to a single label "NONE", so entity type would not be predicted in our model. On ACE05, ACE04, ADE and SciERC, we assess our model under **exact match** where both head and tail of an entity are annotated. For ADE and ACE04, 10-fold and 5-fold cross validation are used to evaluate the model respectively, and 15% of the training set is used to construct the development set. For evaluation metrics, we report F1 scores in both NER and RE. In NER, an entity is seen as correct only if its type and boundary are correct. In RE, A triple is correct only if the types, boundaries of both entities and their relation type are correct. In addition, we report Macro-F1 score in ADE and Micro-F1 score in other datasets.

We choose our model parameters based on the performance in the development set (the best average F1 score of NER and RE) and report the results on the test set. More details of hyper-parameters can be found in Appendix B

5.2 Main Result

Table 1 shows the comparison of our model with existing approaches. In partially annotated datasets WebNLG and NYT, under the setting of BERT. For RE, our model achieves 1.7% improvement in WebNLG but performance in NYT is only slightly better than previous SOTA TpLinker (Wang et al., 2020b) by 0.5% margin. We argue that this is because NYT is generated with distant supervision, and annotation for entity and relation are often incomplete and wrong. Compared to TpLinker, the strength of our method is to reinforce two-way interaction between entity and relation. However, when dealing with noisy data, the strength might be counter-productive as error propagation between both tasks is amplified as well.

For NER, our method shows a distinct advantage over baselines that report the figures. Compared to Casrel (Wei et al., 2020), a competitive method, our F1 scores are 2.3%/2.5% higher in NYT/WebNLG. This proves that exposing relation information to

Method	NER	RE
NYT △		
CopyRE (Zeng et al., 2018)	86.2	58.7
GraphRel (Fu et al., 2019)	89.2	61.9
CopyRL (Zeng et al., 2019)	-	72.1
Casrel (Wei et al., 2020) [†]	(93.5)	89.6
TpLinker (Wang et al., 2020b) [†]	-	91.9
PFN[†]	**95.8**	**92.4**
WebNLG △		
CopyRE (Zeng et al., 2018)	82.1	37.1
GraphRel (Fu et al., 2019)	91.9	42.9
CopyRL (Zeng et al., 2019)	-	61.6
Casrel (Wei et al., 2020) [†]	(95.5)	91.8
TpLinker (Wang et al., 2020b) [†]	-	91.9
PFN[†]	**98.0**	**93.6**
ADE ▲		
Multi-head (Bekoulis et al., 2018b)	86.4	74.6
Multi-head + AT (Bekoulis et al., 2018a)	86.7	75.5
Rel-Metric (Tran and Kavuluru, 2019)	87.1	77.3
SpERT (Eberts and Ulges, 2019) [†]	89.3	79.2
Table-Sequence (Wang and Lu, 2020) [‡]	89.7	80.1
PFN[†]	89.6	80.0
PFN[‡]	**91.3**	**83.2**
ACE05 △		
Structured Perceptron (Li and Ji, 2014)	80.8	49.5
SPTree (Miwa and Bansal, 2016)	83.4	55.6
Multi-turn QA (Li et al., 2019) [†]	84.8	60.2
Table-Sequence (Wang and Lu, 2020) [‡]	89.5	64.3
PURE (Zhong and Chen, 2021) [‡]	**89.7**	65.6
PFN[‡]	89.0	**66.8**
ACE04 △		
Structured Perceptron (Li and Ji, 2014)	79.7	45.3
SPTree (Miwa and Bansal, 2016)	81.8	48.4
Multi-turn QA (Li et al., 2019) [†]	83.6	49.4
Table-Sequence (Wang and Lu, 2020) [‡]	88.6	59.6
PURE (Zhong and Chen, 2021) [‡]	88.8	60.2
PFN[‡]	**89.3**	**62.5**
SciERC △		
SPE (Wang et al., 2020a) [§]	**68.0**	34.6
PURE (Zhong and Chen, 2021) [§]	66.6	35.6
PFN[§]	66.8	**38.4**

Table 1: Experiment results on six datasets. [†], [‡] and [§] denotes the use of BERT, ALBERT and SCIBERT(Devlin et al., 2019; Lan et al., 2020; Beltagy et al., 2019) pre-trained embedding. △ and ▲ denotes the use of micro-F1 and macro-F1 score. NER results of Casrel are its reported average score of head and tail entity. Results of PURE are reported in single-sentence setting for fair comparison.

NER, which is not present in Casrel, leads to better performance in entity recognition.

Furthermore, our model demonstrates strong

performance in fully annotated datasets ADE, ACE05, ACE04 and SciERC. For ADE, our model surpasses table-sequence (Wang and Lu, 2020) by 1.6%/3.1% in NER/RE. For ACE05, our model surpasses PURE (Zhong and Chen, 2021) by 1.2% in RE but results in weaker performance in NER by 0.7%. We argue that it could be attributed to the fact that, unlike the former three datasets, ACE05 contains many entities that do not belong to any triple. Thus utilizing relation information for entity prediction might not be as fruitful as that in other datasets (PURE is a pipeline approach where relation information is unseen to entity prediction). In ACE04, our model surpasses PURE by 0.5%/2.3% in NER/RE. In SciERC, our model surpasses PURE by 0.2%/2.8% in NER/RE. Overall, the performance of our model shows remarkable improvement against previous baselines.

5.3 Ablation Study

In this section, we take a closer look and check the effectiveness of our framework in relation extraction concerning five different aspects: number of encoder layer, bidirectional versus unidirectional, encoding scheme, partition granularity and decoding strategy.

Number of Encoder Layers Similar to recurrent neural network, we stack our partition filter encoder with an arbitrary number of layers. Here we only examine frameworks with no more than three layers. As shown in table 2, adding layers to our partition filter encoder leads to no improvement in F1-score. This shows that one layer is good enough for encoding task-specific features.

Bidirection Vs Unidirection Normally we need two partition filter encoders (one in reverse order) to model interaction between forward and backward context. However, as discussed in section 4.2, our model replaces the backward encoder with a global representation to let future context be visible to each word, achieving a similar effect with bidirectional settings. In order to find out which works best, we compare these two methods in our ablation study. From table 2, we find that unidirectional encoder with global representation outperforms bidirectional encoder without global representation, showing that global representation is more suitable in providing future context for each word than backward encoder. In addition, when global representation is involved,

Ablation	Settings	P	R	F
Layers	N=1	**40.6**	**36.5**	**38.4**
	N=2	39.9	35.7	37.7
	N=3	40.0	36.2	38.0
Bidirection Vs Unidirection	Unidirection (w/o gl.)	**40.6**	**36.5**	**38.4**
	Bidirection (w/o gl.)	40.5	34.6	37.3
		40.4	36.2	38.2
		39.9	35.3	37.5
Encoding Scheme	Joint	**40.6**	**36.5**	**38.4**
	Sequential	40.0	34.2	36.9
	Parallel	36.0	34.4	35.1
Partition Granularity	Fine-grained	**40.6**	**36.5**	**38.4**
	Coarse	39.3	35.5	37.3
Decoding Strategy	Universal	**40.6**	**36.5**	**38.4**
	Selective	38.5	36.3	37.4

Table 2: Ablation study on SciERC. P, R and F represent precision, recall and F1 relation scores. The best results are marked in bold. gl. in the second experiment is short for global representation.

unidirectional encoder achieves similar result in F1 score compared to bidirectional encoder, indicating that global representation alone is enough in capturing semantics of future context.

Encoding Scheme We replace our partition filter encoder with two LSTM variants to examine the effectiveness of our encoder. In the parallel setting, we use two LSTM encoders to learn task-specific features separately, and no interaction is allowed except for sharing the same input. In the sequential setting where only one-way interaction is allowed, entity features generated from the first LSTM encoder is fed into the second one to produce relation features. From table 2, we observe that our partition filter outperforms LSTM variants by a large margin, proving the effectiveness of our encoder in modelling two-way interaction over the other two encoding schemes.

Partition Granularity Similar to (Shen et al., 2019), we split neurons into several chunks and perform partition within each chunk. Each chunk shares the same entity gate and relation gate. Thus partition results for all chunks remain the same. For example, with a 300-dimension neuron set, if we split it into 10 chunks, each with 30 neurons, only two 30-dimension gates are needed for neuron partition. We refer to the above operation as coarse partition. In contrast, our fine-grained partition can be seen as a special case as neurons are split into only one chunk. We compare our fine-grained partition (chunk size = 300) with coarse partition

Dataset	Entity Type	P	R	F	Ratio
ACE05	Total	89.3	88.8	89.0	1.00
	In-triple	95.9	92.1	94.0	0.36
	Out-of-triple	85.8	86.9	86.3	0.64
	Diff	10.1	5.2	7.7	-
ACE04	Total	89.1	89.6	89.3	1.00
	In-triple	94.3	91.2	92.7	0.71
	Out-of-triple	87.1	89.2	88.1	0.29
	Diff	7.2	3.0	4.6	-
SciERC	Total	64.8	69.0	66.8	1.00
	In-triple	78.0	71.1	74.4	0.78
	Out-of-triple	38.9	61.7	47.8	0.22
	Diff	39.1	9.4	26.6	-

Table 3: NER Results on different entity types. Entities are split into two groups: In-triple and Out-of-triple based on whether they appear in relational triples or not. Diff is the performance difference between In-triple and Out-of-triple. Ratio is number of entities of given type divided by number of total entities in the test set (train, dev and test set combined in ACE04). Results of ACE04 are averaged over 5-folds

(chunk size = 10). Table 2 shows that fine-grained partition performs better than coarse partition. It is not surprising as in coarse partition, the assumption of performing the same neuron partition for each chunk might be too strong for the encoder to separate information for each task properly.

Decoding Strategy In pipeline-like methods, relation prediction is performed on entities that the system considers as valid in their entity prediction. We argue that a better way for relation prediction is to take into account all the invalid word pairs. We refer to the former strategy as selective decoding and the latter one as universal decoding. For selective decoding, we only predict the relation scores for entities deemed as valid by their entity scores calculated in the NER unit. Table 2 shows that universal decoding, where all the negative instances are included, is better than selective decoding. Apart from mitigating error propagation, we argue that universal decoding is similar to contrastive learning as negative instances helps to better identify the positive instances through implicit comparison.

6 Effects of Relation Signal on Entity Recognition

It is a widely accepted fact that entity recognition helps in predicting relations, but the effect of relation signals on entity prediction remains divergent among researchers.

Through two auxiliary experiments, we find that the absence of relation signals has a considerable bearing on entity recognition.

6.1 Analysis on Entity Prediction of Different Types

In Table 1, NER performance of our model is consistently better than other baselines except for ACE05 where the performance falls short with a non-negligible margin. We argued that it could be attributed to the fact that ACE05 contains many entities that do not belong to any triples.

To corroborate our claim, in this section we try to quantify the performance gap of entity prediction between entities that belong to certain triples and those that have no relation with other entities. The former ones are referred to as In-triple entities and the latter as Out-of-triple entities. We split the entities into two groups and test the NER performance of each group in ACE05/ACE04/SciERC. In NYT/WebNLG/ADE, since Out-of-triple entity is non-existent, evaluation is not performed on these datasets.

As is shown in table 3, there is a huge gap between In-triple entity prediction and Out-of-triple entity prediction, especially in SciERC where the diff score reaches 26.6%. We argue that it might be attributed to the fact that entity prediction in SciERC is generally harder given that it involves identification of scientific terms and also the average length of entities in SciERC are longer. Another observation is that the diff score is largely attributed to the difference of precision, which means that without guidance from relational signal, our model tends to be over-optimistic about entity prediction.

In addition, compared to PURE (Zhong and Chen, 2021) we find that the overall performance of NER is negatively correlated with the percentage of out-of-triple entities in the dataset. especially in ACE05, where the performance of our model is relatively weak, over 64% of the entities are Out-of-triple. This phenomenon is a manifest of the weakness in joint model: Joint modeling of NER and RE might be somewhat harmful to entity prediction as the inference patterns of In-triple and Out-of-triple entity are different, considering that the dynamic between relation information and entity prediction is different for In-triple and Out-of-triple entity.

Model	ConcatSent		CrossCategory		EntTypos		OOV		SwapLonger		Average
	Ori → Aug	Decline	Ori → Aug	Decline	Ori → Aug	Decline	Ori → Aug	Decline	Ori → Aug	Decline	Decline
BiLSTM-CRF	83.0→82.2	0.8	82.9→43.5	39.4	82.5→73.5	9.0	82.9→64.2	18.7	82.9→67.7	15.2	16.6
BERT-base(cased)	87.3→86.2	1.1	87.4→48.1	39.3	87.5→83.1	4.1	87.4→79.0	8.4	87.4→82.1	5.3	11.6
BERT-base(uncased)	88.8→88.7	**0.1**	88.7→46.0	42.7	89.1→83.0	6.1	88.7→74.6	14.1	88.7→78.5	10.2	14.6
TENER	84.2→83.4	0.8	84.7→39.6	45.1	84.5→76.6	7.9	84.7→51.5	33.2	84.7→31.1	53.6	28.1
Flair	85.5→85.2	0.3	84.6→44.9	39.7	86.1→81.5	4.6	84.6→81.3	**3.3**	84.6→73.1	11.5	11.9
PFN	89.1→87.9	1.2	89.0→80.5	**8.5**	89.6→86.9	**2.7**	89.0→80.4	8.6	89.0→84.3	**4.7**	**5.1**

Table 4: Robustness test of NER against input perturbation in ACE05, baseline results and test files are copied from `https://www.textflint.io/`

6.2 Robustness Test on Named Entity Recognition

We use robustness test to evaluate our model under adverse circumstances. In this case, we use the domain transformation methods of NER from (Wang et al., 2021). The compared baselines are all relation-free models, including BiLSTM-CRF (Huang et al., 2015), BERT (Devlin et al., 2019), TENER (Yan et al., 2019) and Flair-Embeddings (Akbik et al., 2019). Descriptions of the transformation methods can be found in Appendix D

From table 4, we observe that our model is mostly more resilient against input perturbations compared to other baselines, especially in the category of CrossCategory, which is probably attributed to the fact that relation signals used in our training impose type constraints on entities, thus inference of entity types is less affected by the semantic meaning of target entity itself, but rather the (relational) context surrounding the entity.

6.3 Does Relation Signal Helps in Predicting Entities

Contrary to what (Zhong and Chen, 2021) has claimed (that relation signal has minimal effects on entity prediction), we find several clues that suggest otherwise. First, in section 6.1, we observe that In-triple entities are much more easier to predict than Out-of-triple entities, which suggests that relation signals are useful to entity prediction. Second, in section 6.2, we perform robustness test in NER to evaluate our model's capability against input perturbation. In the robustness test we compare our method - the only joint model to other relation-free baselines. The result suggests that our method is much more resilient against adverse circumstances, which could be (at least partially) explained by the introduction of relation signals. To sum up, we find that relation signals do have non-negligible effect on entity prediction.

The reason for (Zhong and Chen, 2021) to conclude that relation information has minimal influence on entity prediction is most probably due to selective bias, meaning that the evaluated dataset ACE05 contains a large proportion of Out-of-triple entities (64%), which in essence does not require any relation signal themselves.

7 Conclusion

In this paper, we encode task-specific features with our newly proposed model: Partition Filter Network in joint entity and relation extraction. Instead of extracting task-specific features in a sequential or parallel manner, we employ a partition filter encoder to generate task-specific features jointly in order to model two-way inter-task interaction properly. We conduct extensive experiments on six datasets to verify the effectiveness of our model. Overall experiment results demonstrate that our model is superior to previous baselines in entity and relation prediction. Furthermore, dissection on several aspects of our model in ablation study sheds some light on what works best in our framework. Lastly, contrary to what previous work has claimed, our auxiliary experiments suggest that relation prediction is contributory to named entity prediction in a non-negligible way.

8 Acknowledgements

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019.

FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018a. Adversarial training for multi-context joint entity and relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2830–2836, Brussels, Belgium. Association for Computational Linguistics.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018b. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 551–560, Portland, Oregon, USA. 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.

Dennis Diefenbach, Vanessa Lopez, Kamal Singh, and Pierre Maret. 2018. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information systems*, 55(3):529–569.

Markus Eberts and Adrian Ulges. 2019. Span-based joint entity and relation extraction with transformer pre-training. *arXiv preprint arXiv:1909.07755*.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418, Florence, Italy. Association for Computational Linguistics.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547, Osaka, Japan. The COLING 2016 Organizing Committee.

Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. *Journal of Emerging Technologies in Web Intelligence*, 2:258–268.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of biomedical informatics*, 45(5):885–892.

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. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 917–928, Vancouver, Canada. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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 *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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

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

of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1858–1869, Doha, Qatar. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 148–163. Springer.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

Yongliang Shen, Xinyin Ma, Yechun Tang, and Weiming Lu. 2021. A trigger-sense memory flow framework for joint entity and relation extraction. In Proceedings of the Web Conference 2021, pages 1704–1715.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958.

Tung Tran and Ramakanth Kavuluru. 2019. Neural metric learning for fast end-to-end relation extraction. arXiv preprint arXiv:1905.07458.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. Linguistic Data Consortium, Philadelphia, 57:45.

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

2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1706–1721, Online. Association for Computational Linguistics.

Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, Yicheng Zou, Xin Zhou, Jiacheng Ye, Yongxin Zhang, Rui Zheng, Zexiong Pang, et al. 2021. Textflint: Unified multilingual robustness evaluation toolkit for natural language processing.

Yijun Wang, Changzhi Sun, Yuanbin Wu, Junchi Yan, Peng Gao, and Guotong Xie. 2020a. Pre-training entity relation encoder with intra-span and inter-span information. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1692–1705, Online. Association for Computational Linguistics.

Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. 2020b. TPLinker: Single-stage joint extraction of entities and relations through token pair linking. In Proceedings of the 28th International Conference on Computational Linguistics, pages 1572–1582, Barcelona, Spain (Online). International Committee on 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.

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. 2019. Tener: adapting transformer encoder for named entity recognition. arXiv preprint arXiv:1911.04474.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In Coling 2010: Posters, pages 1399–1407, Beijing, China. Coling 2010 Organizing Committee.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002), pages 71–78. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Xiangrong Zeng, Shizhu He, Daojian Zeng, Kang Liu, Shengping Liu, and Jun Zhao. 2019. Learning the extraction order of multiple relational facts

Dataset	#Sentences			$\|\mathcal{E}\|$	$\|\mathcal{R}\|$
	Train	Dev	Test		
NYT	56,195	5,000	5,000	-	24
WebNLG	5,019	500	703	-	170
ADE	4,272 (10-fold)			2	1
ACE05	10,051	2,424	2,050	7	6
ACE04	8,683 (5-fold)			7	6
SciERC	1,861	275	551	6	7

Table 5: Statistics of datasets. $\|\mathcal{E}\|$ and $\|\mathcal{R}\|$ are numbers of entity and relation types. In NYT and WebNLG, entity type information is not annotated.

in a sentence with reinforcement learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 367–377, Hong Kong, China. Association for Computational Linguistics.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514, Melbourne, Australia. Association for Computational Linguistics.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, Vancouver, Canada. Association for Computational Linguistics.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *North American Association for Computational Linguistics (NAACL)*.

A   Dataset

We evaluate our model on six datasets. NYT (Riedel et al., 2010) is sampled from New York Times news articles and annotated by distant supervision. WebNLG is originally created for Natural Language Generation task and is applied by (Zeng et al., 2018) as a relation extraction dataset. ACE05 and ACE04 (Walker et al., 2006) are collected from various sources, including news articles and online forums. ADE (Gurulingappa et al., 2012) contains medical descriptions of adverse effects of drug use. SciERC (Luan et al., 2018) is collected from 500 AI paper abstracts originally used for scientific knowledge graph

construction. Following previous work, we filter out samples containing overlapping entities in ADE, which makes up only 2.8% of the whole dataset. Statistics of the datasets can be found in table 5

B   Implementation Details

We leverage pre-trained language models as our embedding layer. Following previous work, the versions we use are *bert-base-cased*, *albert-xxlarge-v1* and *scibert-scivocab-uncased*. Batch size and learning rate are set to be 4/20 and 1e-5/2e-5 for SciERC/Others respectively. In order to prevent overfitting, dropout (Srivastava et al., 2014) is used in our word embedding, entity span and triple representation of task units (set to be 0.1). We use Adam (Kingma and Ba, 2015) to optimize our model parameters and train our model for 100 epochs. Also, to prevent gradient explosion, gradient clipping is applied during training.

C   Analysis on Overlapping Pattern and Triple Number

For more comprehensive evaluation, we assess our model on NYT/WebNLG datasets on different triple overlapping patterns (see section 2 for the detailed description of these patterns) and sentences containing a different number of triples. Since previous work does not compare triple overlapping pattern and triple number in ADE/ACE05/ACE04/ScIERC given that EPO triples are non-existent in these datasets, comparison result is not included for these datasets.

As is shown in figure 3, Our model is mostly superior to the other two baselines in all three categories. Interestingly in normal class, our model performs significantly better in WebNLG, but the score in NYT is basically on par with TpLinker. We argue that this could probably be caused by the fact that NYT, generated by distant supervision, is much more noisier than WebNLG. Besides, sentences of normal triples are likely to be much noisier than sentences of EPO and SEO triples since there is a higher chance for incomplete annotation. Thus it is unsurprising that no significant improvement is achieved in predicting normal triples of NYT.

Besides, from figure 4 we observe that our model performs better in sentences with more than five triples on both datasets, where interaction between entity and relation becomes very complex. The strong performance in those sentences confirms the
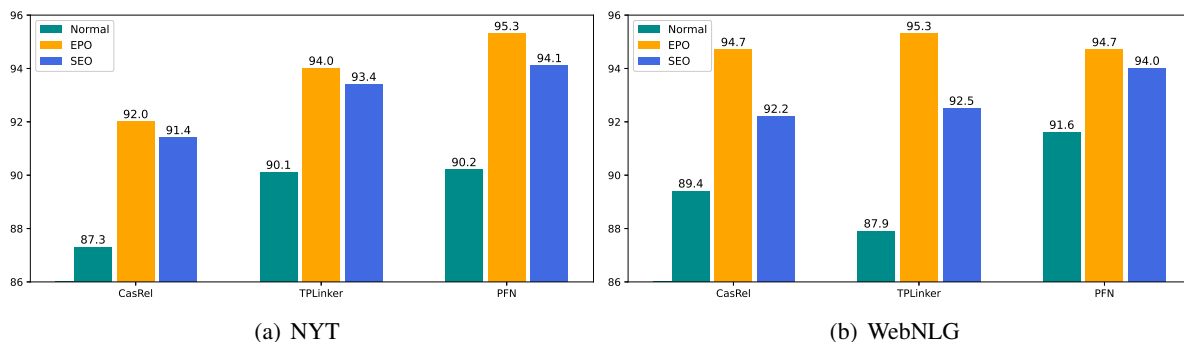
(a) NYT

(b) WebNLG

Figure 3: F1-score of relation triple extraction on sentences with three different overlapping patterns.
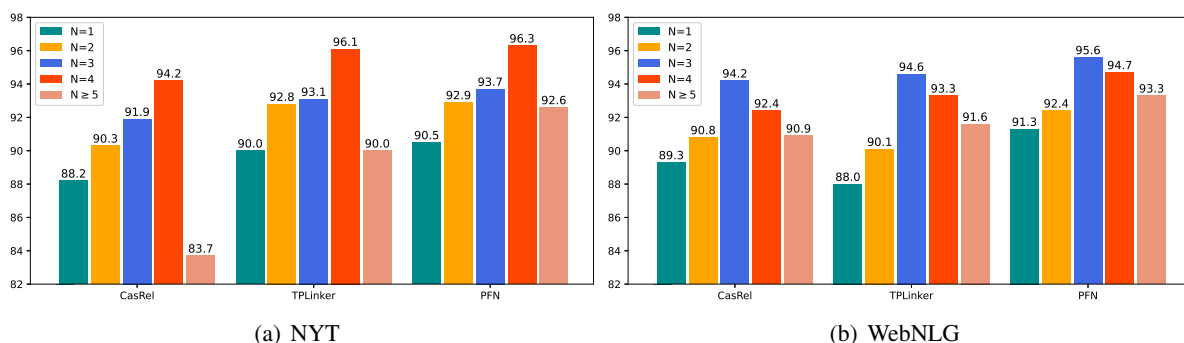

(a) NYT

(b) WebNLG

Figure 4: F1-score of relational triple extraction on sentences containing N triples, with N ranges from 1 to ≥ 5.

superiority of our model against other baselines.

D Details of Robustness Test

Descriptions of the transformation methods used in Table 4 are listed as follows:

1. ConcatSent - Concatenate sentences to a longer one.

2. CrossCategory - Entity Swap by swaping entities with ones that can be labeled by different types.

3. EntTypos - Swap/delete/add random character for entities.

4. OOV - Entity Swap by out-of-vocabulary entities.

5. SwapLonger - Substitute short entities for longer ones.

Transformations of RE are not viable for the following reasons:

1. The input is restricted to one triple per sentence.

2. The methods include entity swap, which is already covered in NER.

3. The methods include relation-specific transformations (Age, Employee, Birth) and ACE05 does not have these type of relations.

4. The methods include inserting descriptions of entities, which is unfair because it might introduce new entity and relation.