GENERATE TRIGGERS IN NEURAL RELATION EXTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

In the relation extraction task, the relationship between two entities is determined by some specific words in their source text. These words are called relation triggers, which are the evidence to explain the relationship; other words are called irrelevant words. The current relationship extraction neural network model aims at identifying the relation type between two entities mentioned in source text by encoding the text and entities. However, these models cannot output the relation triggers, but only gives the result of relation classification. Although models can generate weights for every single word through the improvement of attention mechanism, the weights will be affected by irrelevant words essentially, which are not required by the relation extraction task. In order to output relation triggers accurately, we propose a novel training framework for Relation Extraction (RE) that reduces the negative effect of irrelevant words on them in the encoding stage. In specific, we leverage Evolutive Mask based Point Network (EMPN) as a decoder to generate relation triggers and encode these words again. For an ordered output in relation triggers, we utilize order loss to constrain the output order in them. Extensive experiment results demonstrate that the effectiveness of our proposed model achieves state-of-the-art performance on three RE benchmark datasets.

1 INTRODUCTION

Relation Extraction (RE) is a common task in Information Extraction (IE). The semantic relationships in text between two entity mentions, head entity and tail entity, rely on a part of words (Shen et al., 2021). Those words provide the basis for scientific judgment of relation type. Given a sentence: *Our experiments on real datasets show that the resulting detector is more robust to the choice of training examples, and substantially...*, and the entities: real datasets; detector, the task is to detect that they coincide with predefined relation type *USED-FOR*. Although the sentence sample is incomplete, it can tell the type by inferring result from words or phrases: *on, show, resulting, training examples*.

To improve the relevance of relation extraction results and the most likely information, recent methods have been proposed. Some rely on vector functional expression (**?**), distance supervised relation sets (Ye & Luo, 2020), and auxiliary information which is irrelevant to a word vector (e.g., syntax-tree (Veyseh et al., 2020) or webpage snippets (Lockard et al., 2020)). Compared with the only using of word and feature embeddings, these methods ask for extra information instead of extracting from the text. Others tend to reconstitute encoding order as a strategy for gathering relation triggers and increasing identification probability of these parts in training, such as employing GCN (Guo et al., 2020) and Attention (Guo et al., 2019). Even if these two methods can give each word a weight, they cannot bring out an effective screening, causing a high probability that low weight word vectors still propagate but not to cease their work. Thus, these methods cannot extract the relation triggers accurately. For the sake of getting them and improving relation extraction performance, the overall method is to generate candidate words as triggers and reuse them for classification.

(Jiang et al., 2020) suggest that language model contains a lot of knowledge, which can supplement the blank words in a text by template and specific relationship. We are inspired by this view and turn to generative model. In general, relation triggers come from the sentence they lie, without other texts. Therefore, the result space of the generative model is the original text. We view the generative model as a self-sampling sequence decoder model, which coincides with Pointer Network (Vinyals et al., 2015). The way it gets the prediction result is to output probability distributions that can be thought pointers to the location of input words. We encode the entity mentions and original text with LSTM and decode them with Pointer Network, to get the positions. Overall, after these pointers converge to the position of relation trigger word in the training stage, they can be employed as the judgment of relation extraction by concatenated with entity mentions. The concatenated vectors will be mapped to the predefined classification types.

The above thinking shows a path to get relation triggers and improve the effect of relation extraction, but leaves the following issues to be solved:

1) Most of relation extraction datasets do not contain relation triggers. Relation extraction datasets focus on source text, entity types, and relation types, which support recent studies to do the research without extra data (Bai et al., 2020; Alt et al., 2020; Ni et al., 2020). It involves a lot of labor and time to annotate trigger words in the dataset. So, we transfer the thought from Weakly Supervised Learning (Zhou, 2018) to Relative Supervised Learning, for the unlabeled span of the texts sake. The Weakly Supervised Learning allows the incomplete, inexact, and inaccurate supervision data, at least a small or tiny amount of supervision data but not zero. The supervision data with the task of relation extraction links to relation types, not relation triggers. Thus, we supervise the triggers with the help of relation types. We make sure the triggers share the error with relation extraction in the training stage, because we combine them with the output of entity mentions.

2) Attention based method cannot stop the low weight vector propagation in neural network. Attention based method assigns different weights for segregated morphemes in a sentence. The low weight morphemes are considered interfering with prediction and should be abandoned. However, the method does not realize it, but allow them to participate in subsequent process. This problem still exists in recent work (Shahbazi et al., 2020; Li et al., 2020; Qu et al., 2020). Although (Li et al., 2020) have utilized PCNN and parallel manner to capture a global sentence representation, the improved range of their experiment suffers. Therefore, we propose a new framework that we re-encode the relation triggers by second LSTM encoder with the aim of eliminating interference in getting hidden states. We stop employing all the hidden states that output from the first one. But we retain entity mention parts and the relation trigger outputs to enhance these parts features.

3) Without supervision, there is a strong possibility that Pointer Network will generate identical probability distribution result which selects the same words in text. The results of Pointer Network reflect the models attention to each element in source text. The higher the probability is, the more attention the words get. Nevertheless, the higher parts may be always the higher in subsequent probability distribution, which is not allowed due to words that determine the relation type and are not always one. (See et al., 2017) join Coverage Mechanism into their model for encouraging more attention to the previously unnoticed words. This method is based on supervision results so that it is not fit for our aims. The lack of supervision triggers will not let coverage mechanism work. Because of this, we propose a new Growth Mechanism of location mask. Combined with Pointer Network, we name it Evolutive Mask based Point Network (EMPN). Once Pointer Network predicts a new word location, the mask of the network will shield the location and prevent it from being selected next time. We use the hidden states of head entity and tail entity to finish this process in order to establish a connection between masks and entities.

By overcoming three problems, we present a novel training framework. And our contributions are as follows: (1) We propose Relative Supervised Learning method that supervises non-labelled text spans from annotated result. (2) We propose a new approach that re-encodes the selected word embeddings to ignore the interfering words. (3) We propose a new mask growth mechanism that allows masks to lock the value of select locations before, and we name it as Evolutive Mask based Point Network (EMPN). We conduct evaluations on three benchmark datasets (i.e., SciERC, Conll2004, and Wiki80) which demonstrate the effectiveness of our RE framework for exploring the challenges of extracting relation triggers.

2 RELATED MODEL

As a sequence decoder, Pointer Network neednt to map its output to preset glossary, but an exact length of probability distribution. Thats the reason it can duplicate the words from the input. According to this, it is suitable for locating task with supervision data. Thus, Pointer Network has been employed in article summarization (Anh & Trang, 2019; Sun et al., 2018), neural answer framework (Gu et al., 2016; Liu et al., 2021), sentences ordering (Yin et al., 2020), aspect extraction (Wei et al., 2020), and argumentative structure features extraction (Kuribayashi et al., 2017). Similar with the summarization, the triggers that we extract are a part of source text.

In relation extraction problem, Pointer Network serves as location identification model, especially the location of entities in joint extraction (Katiyar & Cardie, 2017; Nayak & Ng, 2020; Roth et al., 2018). This feature of the Network is applied for more complex research direction in relation extraction. (Park & Kim, 2019) employ two Pointer Network to solve 1-to-n head-rail relations. They let the first one finds relations from one head entity to several tail entities, and the second one reverses the head and tail. (Pang et al., 2019) make Pointer Network generate three probability distribution matrices to predict head pointer, offset, and relation. However, these methods ignore the relation triggers that we need. The absence of annotated triggers urges us to convert the direct supervision to the relative supervision.

Generation Model has been used in relation extraction problem in recent years. (Shen et al., 2021) propose to extract entity and relation triggers to rationalize the prediction. But their method doesnt match our demand or solve the problems in section one. The method cannot output the select triggers because the Memory-Aware method they propose cannot specify locations of sentence spans. (Kilicoglu et al., 2020) generate reference words with the hand of a broad-coverage NLP system called SemRep. (Zhang et al., 2020) generate the embeddings for the positions of the triggers and entity relations and concatenate with basic linguistic features. All of them give ambiguous represention of relation triggers. To our knowledge, we are the first group to generate trigger words and output them in RE research works.

3 APPROACH

We consider the RE problem as multi-category problem and output the select word sequence at the same time. $f(x) = a + b$For source text, we adopt a similar input method in the work of (Veyseh et al. 2020) that we concatenate information embeddings with word embeddings as needed. Let $S = (w_1, w_2, \ldots, w_i, \ldots, w_j, \ldots, w_n)$ denotes a source text of length n that contains entity mentions at location i and j. w_k represents the k position word vector concatenated with the three following vectors: (a) the pretrained word embeddings of word k, with pretrained embedding table from Glove.6B 50 dimensions (Pennington et al., 2014). (b) two relative position embeddings from k to i and k to j. The absolute values of distance from word k to entity mention give this word locations in training. (c) the entity type embeddings if they exist. Some datasets provide entity types like BIOE labels for entity mentions (such as SciERC). For entity mentions, we just input word embeddings as (a) in source text part, name as w_s and w_o.

Our RE framework can be divided into four parts: basic encode layer, location decode layer, re-encode layer and output layer, which are depicted in Figure 1. Basic encode layer contains text LSTM encoder and entity LSTM encoder, transforming S, w_s and w_o to three vectors: Q_S, h_s and h_o, represent sequence output of S, final hidden states of w_s and w_o after encoding. We allocate entity masks m_i and m_j for Q_S to extract hidden states from location i, j, and -1: H_i, H_j, and fh, to be one of relation determinant. Location decode layer only contains EMPN. It receives h_s as decoding input, h_o as attention supervision input, final hidden state H_n and final cell state C_n of the text LSTM encoder as state inputs. EMPN will output the probability distributions a and select trigger sequence $L = (w_k, w_m, \ldots, w_p)$ with length e at once. Re-encode layer contains an LSTM encoder and a FFNN. We prepare the former for re-encoding L and convert it to encoded sequence U. And the latter maps the w_s and w_o to f_s and f_o, with the same dimension d with the output of the former. Finally, we concatenate f_s, f_o, H_i, H_j, L, U, fh and get the relation type p by a nonlinear activation function in output layer. We also supervise a to rationalize sequence L.

We will detail all the layers in the following subsections. All main variables for the sections formulae have been listed above.

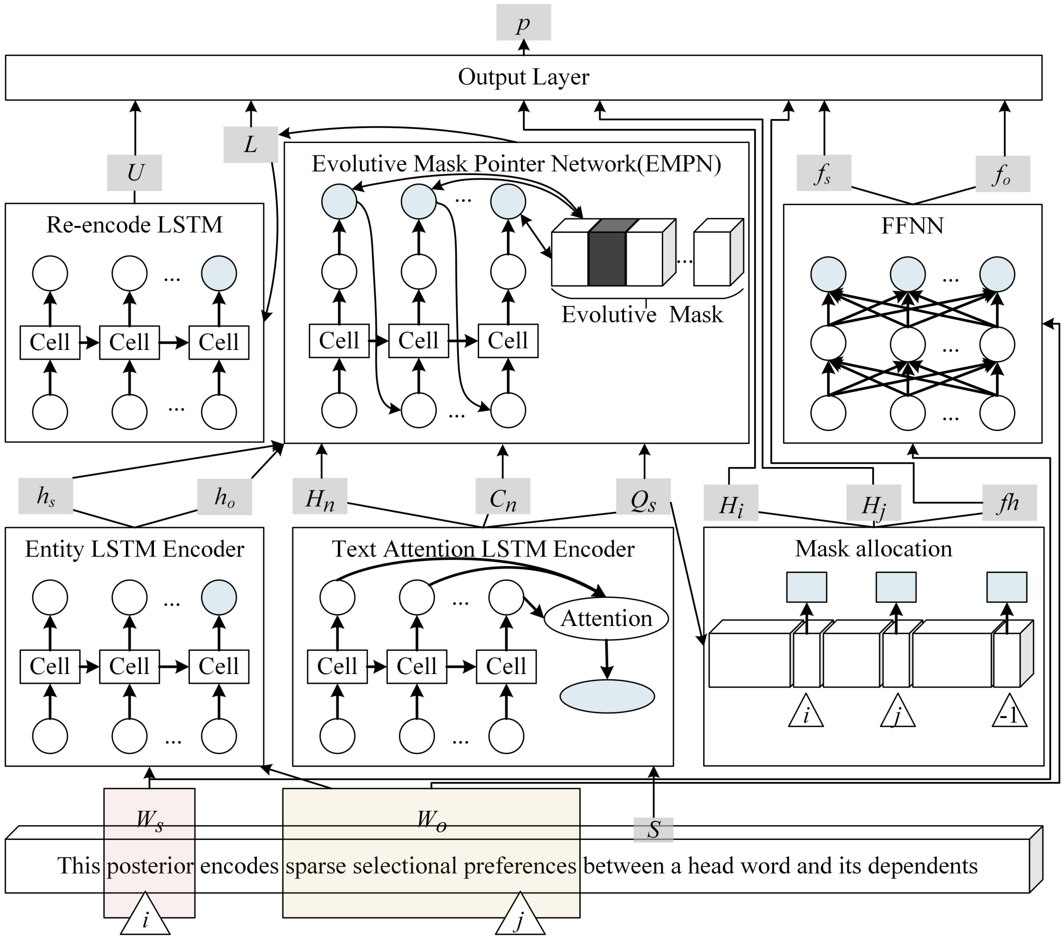

Figure 1: RE Framework

3.1 BASIC ENCODE LAYER

LSTM encoders: The aim of basic encoder layer is to get vector representations of source text, the head entity and the tail entity. We use two different LSTM, Entity LSTM Encoder and Text Attention LSTM Encoder, to encode entity and text for getting different information level granularity features. The source text implicates sentence or article level information, more abundant than word level information in entity mentions. Nonetheless, after encoding, the representations of source text will weaken the information of some of its words because of the forget gate in LSTM. The representations of them complement each other by this way. This process can be described as follows:

$$Q_s = lstm_1(S) \tag{1}$$

$$h_s = lstm_2(w_s, h_{t-1}) \tag{2}$$

$lstm_1$ denotes text LSTM encoder and $lstm_2$ denotes entity LSTM encoder. The head and tail share the same. h_{t-1} is the last time step before the generation of h_s. For calculation, the Q_S is sequence but h_s is non-sequence. We deploy self-attention method in (Vaswani et al., 2017) for text LSTM encoder to get dependency information from words to sentence semantics.

Mask allocation: The sequence output of text LSTM encoder Q_S denotes vector representation of all words in source text. But Q_S may contain much interference information from the low value words for our goal. We therefore use masks to filter the outputs at the head and tail locations and let the head and tail locate at position i and j. We first repeat mask m_i and m_j for several times which equals to R^d, the last dimension of Q_S. After that, they multiply by Q_S in elementwise. Finally, we sum each of them in axis 1 and get hidden state H_i and H_j. Because all elements are 0 except for position i and j, we can extract H_i and H_j exactly. This process can be described as follows:

$$rmask_k = repeat(m_k, R^d) \tag{3}$$

$$EH_k = rmask_k * Q_s \tag{4}$$

$$H_i = sum(EH_{k(1)}, EH_{k(2)}, ..., EH_{k(n)}) \tag{5}$$

Note that n is the length of sequence, $k \in \{i, j\}$ clarifies that we extract the hidden state of head or tail. In addition, we also get the final hidden state of Q_S, fh, for representing the semantic of the text.

3.2 LOCATION DECODE LAYER

EMPN is the only model in this layer and output an ordered select vectors for its next layer, can be viewed as a kind of decoder. For a general LSTM decoder, given a hidden state h_{t-1} and cell state c_{t-1} at time $t-1$, it produces h_t at time t for mapping to the result vector space. We describe this process as below:

$$h_t = lstmdecoder(h_{t-1}, c_{t-1}) \tag{6}$$

We modify Pointer Network into EMPN to meet the requirement of outputting triggers. Given the influence of entity mention order and trigger order, we will introduce it in next four topics: decoding input, attention strategy, mask evaluation mechanism, and order consistency.

decoding input: The input of Pointer Network is the same as general decoder. It takes final hidden state of an encoder as input every time step. Different from Pointer Network, we need to consider the entity mentions in the relation extraction task, so we replace H_n by h_s for the first step. For h_o, it is employed by attention part and will be discussed in the next subsection. After the first step, we collect the pointed word vector w_k that generate from probability distributions a and S. It is chosen as the next input. We set a hyperparameter e to indicate how many times the process will loop. By this way, such input takes the entity mentions into account. The following formula shows how word vector is chosen from step time one to e.

$$w_p = S * element\text{–}greater(a, reducemax(a)) \tag{7}$$

w_p is the selected vector from S, and also the input of EMPN in the $p+1$ time. The restrictions of p are $(1 < p \leq e, p \in Z_+)$. $reducemax$ means to keep the max value and set others the same with it. $element\text{–}greater$ means to compare every element in the two matrices and set the same or larger to 1, others to 0.

attention strategy: Attention mechanism is used in Pointer Network to generate probability distributions a. In Supervised Learning mode, it integrates Q_S and Q_L with tanh function and converts them to a with $softmax$ function. Q_L denotes the hidden state sequence of decoder, consisted of every time step hidden state in formula (6). Finally, a will map to result P.

$$a_j^i = softmax(V^T tanh(W_1 H_i^{(s)} + W_2 H_j^{(L)})) \tag{8}$$

$$p\left(o_j = P | o_1, \ldots, o_{j-1}\right) = a_j \tag{9}$$

$H_i^{(s)}$ and $H_j^{(L)}$ denote the i-th and j-th hidden states of text LSTM encoder and Pointer Network. o_j denotes the output of j-th hidden states. W_1 and W_2 denote the weights of integration. V shrinks the tanh result by reducing dimension d to 1. Thus, $softmax$ function works on sentence length dimension to choose vectors from S. We assert that a_j is the largest one in a_j^i $(1 \leq i \leq n)$. So, the pointer a_j links to result P.

However, we modify the value of a by receiving errors from select trigger sequence L in training, hence we couldnt adopt formula (9) as we essentially realize Relative Supervised Learning that supervises triggers by relation type. In EMPN, since we think about entity mentions, we concatenate h_o with every hidden state of EMPN as a new vector instead of $H_j^{(L)}$. According to this, we build a word path from head entity to tail entity with formula (7). The updated calculation method is as follows:

$$a_j^i = softmax(V^T tanh(W_1 H_i^{(s)} + W_2(\begin{matrix} h_o \\ H_j^{(L)} \end{matrix}))) \tag{10}$$

mask evaluation mechanism: Evolutive mask is a key method to generate diversity triggers. Considering the case where we dont employ any mask or only employ entity mask for preventing entity location vectors to be selected, it equates to static mask when generating the probability distributions a. The mask will lose the function that induces EMPN to select a different word with previous time step in the consideration of one or several vectors being always noticed by a. We couldnt set extra loss for relation trigger sequence due to no supervised one. Consequently, we overcome this problem by mask evaluation mechanism.

(Liao et al., 2020) propose variable mask in generative model. The mask on a certain position is drawn from a probabilistic distribution and exchanged by specific word in generation phase. In addition, they prepare a set to mark the pointed positions and ignore them afterwards. We consult these two features of variable mask to design our evolutive mask. In each step, the last mask is updated by adding a new indicator generated from a. The indicator r points one of word vectors in S, the same value with multiplier in formula (7). Let the sum of source text mask m_S, entity mask m_i and m_j be the basic mask. We add r to it for changing the value that in $softmax$ function in formula (10). And in step t, this formula will update into the following:

$$a_{j(t)}^i = softmax(V^T tanh(\varepsilon) + r_t + m_S + m_i + m_j) \tag{11}$$

where $\varepsilon = W_1 H_i^{(s)} + W_2(\begin{matrix} h_o \\ H_j^{(L)} \end{matrix})$.

Algorithm 1 depicts the algorithm to update indicator r. This process begins with h_s as the first input of EMPN and ends until the time step is up to cycle number defined by hyperparameter e. Because the update processes involves other EMPN process above, input method and attention strategy is also added.

By this mechanism, we make sure every word comes from different positions. The evaluated mask will set a low weight for the selected.

order consistency: To make the trigger more reasonable, we leverage a constraint for its sequence order. It refines the triggers into the origin order as source text. We set order loss to represent this constraint in training. For r in the last time step, let r_{front} be the $(r_1, r_2, ..., r_{e-1})$; r_{back} be the $(r_2, r_3, ..., r_e)$. If the trigger is consistent with original order, the element-wise differences between r_{back} and r_{front} are all equal to 1. Let ones be a vector with length $e - 1$ and element 1, the order loss can be calculated as follows:

$$L_{order} = abs(r_{back} - r_{front} - ones) \tag{12}$$

Except for order supervision, it also makes triggers cluster into several blocks. The optimization strategy of order loss is to cut the difference of every neighbor in r to 1. Besides the ideal that all the difference is 1, the value with non-one will separate the trigger into segments which are small parts in source text.

3.3 RE-ENCODE LAYER

For interrupting the effect of irrelevant words, EMPN select some triggers L from source text vector S, which is un-encoded. Therefore, we employ re-encode LSTM encoder to output sequence U, which only contains triggers information. Whatever triggers are outputted from EMPN, the irrelevant words are unable to persuade the re-encode LSTM. So, our framework stops the low weight vector propagation in the model. This can be described as followed:

Table 1: Indicator updating process.

Algorithm 1: update indicator

Input:
word vector of source text: $S = (w_1, w_2, \dots, w_i, \dots, w_j, \dots, w_n)$
text LSTM encoder: Q_S
final hidden states of head and tail entity mentions: h_s, h_o
text LSTM encoder final hidden state: H_n
text LSTM encoder final cell state: C_n
source text mask: m_S
entity masks: m_i, m_j
hyperparameter: e $(e < n)$
Parameter:
EMPN input h_s
EMPN states (H_n, C_n)
basic indicator r 0
Output: r
While time step t is smaller than e **do**:
i)decode the input and states and generate new hidden states and new cell states (h_t, c_t) with formula (6)
ii)let h_t and h_o concatenate to new vector and duplicate it on second dimension, from 1 to n, the same length with S
iii)give name D_S to the result of ii) and calculate probability distributions a with formula (11), and the variables include $D_S, Q_S, m_S, m_i, m_j, r$
iv)get probability distributions a of iii) and calculate new indicator for updating r and the selected word w_t by formula (7)
v)for next turn, update other values: EMPN input w_t, EMPN states (h_t, c_t), t $t + 1$
end while

$$U = lstm_{re-encode}(L) \tag{13}$$

For w_s and w_o, they are not embedded with positions or entity types. We utilize a FFNN to stretch the last dimension to d, consistent with Q_S. There is no active function in order to reserve original vector distribution of them. The outputs are f_s and f_o as follows:

$$f_s = FFNN(w_s) \tag{14}$$
$$f_o = FFNN(w_o) \tag{15}$$

3.4 OUTPUT LAYER

According to above layers, we select six vectors from or related to entity mentions, f_s, f_o, H_i, H_j, L, U, which represent different information levels. We follow the work of (Pouran Ben Veyseh et al., 2019), let C denote the concatenation of these vectors. f_s and f_o are basic entity representations as they come from linear transformations of w_s and w_o. H_i and H_j are entity representations in source text from text LSTM encoder. L and U are trigger representations from EMPN extraction and re-encode LSTM. In output layer, C will be fed into output FFNN with $softmax$ as activate function for predicting relation type p of current sample.

$$pred(y = p|w_s, w_o, S) = FFNN(C) \tag{16}$$

For the loss function of relation type estimation, we adopt multi-class cross entropy. The loss of type can be described as follows:

$$L_{type} = -\frac{1}{n}\sum_k [q_k ln p_k + (1 - q_k) ln(1 - p_k)] \tag{17}$$

k denotes the k-th sample in a batch of num n. q denotes the real type of relations.

The word number of triggers in some samples may surpass 1 while others stay in 1. Related to these cases, we define two types of final loss L_{final}. One is the same with L_{type}, the other is $L_{type} + L_{order}$. The effect will be shown in the next section.

4 EXPERIMENTS

4.1 DATASET

We evaluate our framework with three benchmark datasets: SciERC, Wiki80 and Conll2004. The SciERC dataset contains 500 scientific abstracts for RE task with 350 in training set, 100 in testing set, and 50 in validation (named dev in file) set. Owing that source texts for RE are split from the abstracts, the RE samples in training, testing, and validation are numbers (3219, 974, 455). There are 6 types (*Metric, Method, Generic, Material, OtherScientificTerm*) of entities and 7 types (*CONJUNCTION, FEATURE-OF, EVALUATE-FOR, HYPONYM-OF, COMPARE, USED-FOR, PART-OF*) of relations. They form structures of splicing patterns by 168 types of triples with specific head entity type, relation type, and tail entity type. All these structures aggregate what we call schema.

The Conll2004 dataset contains training (910 articles), testing (288 articles), and validation (243 articles) parts. The samples of them are numbers (1273, 422, 353). There are 3 types (*Location, Organization, People*) of entities and 5 types (*OrgBased-In, Kill, Live-In, Located-In, Work-For*) of relations. We set No Entity Type for Conll2004 dataset as the entity type will leak the relation label by schema.

The Wiki80 dataset only involves training and testing parts. Different from the above, the form of it is independent samples within tokens, head entity, tail entity, and relation type. The training involves 50400 samples which contain 80 types of relation. The testing involves 5600 samples. It is certainly balanced dataset as each type of relations contains 630 samples in training part and 70 samples in testing part. Note that Wiki80 doesnt contain entity type, so we mark the entity type with *subj* and *obj*, representing head entity and tail entity.

4.2 EXPERIMENT SETUP

Parameter Settings: We implement all models in our framework with Keras 2.4.3 and Tensorflow 2.2.0. The lengths of every embedding are: 50 for word, 20 for entity type, 100 for relative location. Output for all encoders and EMPN is 170, equal to the sum of all embeddings. We employ RMSprop optimizer and set 0.001 to learning rate. The batch size of one turn is 128. We map out our iteration stopping strategy from (Yin et al., 2019)s work. If the effect of RE turns out to be no better within more than 40 epochs, we will stop running and record the best one.

Evaluation metrics: We follow the standard evaluation protocol by employing both macro-average F1(Macro-F) and micro-average F1(Micro-F) for all three datasets. Macro-F suits for exhibit the performance of relation categories while Micro-F for the samples. We make sure all prior works generate Macro-F and therefore we use this when comparing.

Implementation Details: In order to compare the variants of our basic framework, we add order loss and zero-one tasks to EMPN to demonstrate different values of decoded length e. The decoded length e changes from 1 to 10 for the sake of discussing how triggers control the performance. The input of sample contains eight fields (*doc_key, sentence_span, current_annotations, ent1_start, ent1_end, ent2_start, ent2_end, entity2, entity1, relation_type*). We regard relation_type as the standard result to correct the prediction from the first seven fields.

We divide the RE into two subtasks, normal RE task and zero-one mixed RE task. Normal RE is source classification task which complies with the number of samples in all datasets. Zero-one mixed RE is to add independent entities with No-relation type to original samples. The entity pairs have identical entity type combinations with original samples except relation types in added samples. We set this to examine whether the result of such samples exhibits the same trend with normal tasks. Wiki80 dataset doesnt contain any articles, leading to no disrelated entity and decline in quantity of No-relation samples.

Baselines: We divide baselines into three types, which represent different coding of words. Note that baselines include Joint models, which combine Entity and Relation Extraction together. This means the F1 value of these baselines contains the situation that two entities which are no relation may be extracted as any type of relation. So, it is equivalent to our zero-one tasks, and we compare these baselines to our zero-one training mode F1 value. We finish the comparation with some baselines from the original papers, and others from reappearance of codes:

i) Joint models: Joint models enable entity hidden states to reuse in relation extraction stage. Baselines here are (Eberts & Ulges, 2020; Bekoulis et al., 2018; Zhong & Chen, 2021; Luan et al., 2018; 2019; Shen et al., 2021).

ii) Sequence models: Sequence models encode vectors to capture the feature of them, such as (Zhang et al., 2017; Veyseh et al., 2020; Zhou et al., 2016). Our framework is a kind of sequence model as we only use LSTMs for RE.

iii) Graph models: Graph models image the words as a tree or graph and extract relations by long distance model (i.e., GCN), such as (Guo et al., 2019; Zhang et al., 2018).

4.3 Explanation of experiment results

For purpose of getting comparisons, we introduce baseline results in our experiments. We choose the summits of every dataset to compare. Table 2 shows normal RE task that the results are compared with other sequence models and graph models. And table 3 shows zero-one mixed RE task because the baselines for it is joint models. We use $e = x$ to represent the trigger length is x. Our framework is named EMPN or 0-1 EMPN in two tables.

According to table 2 and table 3, with and without the only external resource entity types, EMPN achieves the best performance on all datasets in Normal RE. This clearly demonstrates the effectiveness of EMPN in the case of RE. In these datasets, EMPN keeps the gap between 0.6% and 2.2%, outperforming other models. We attribute the improvement to three reasons. First, it is the EMPN in our framework that improves the performance. The encoder in our framework is LSTM, while the counterpart in baselines is improved. Second, the triggers selected by EMPN are different, so they can increase discrimination in different samples with identical origin text. Finally, the output layer allows prediction from ample information layers, comprising trigger and re-encoded vectors.

Our framework tends to optimize trigger choice while these models try to mine more structure and text information, which contributes to the lower Macro-F value in zero-one experiments of Conll2004 dataset. The disrelated entities may generate plenty of No-relation samples and make the dataset unbalanced, which is able to interpret the gap of micro F1 value and Macro-F value. We reset the experiment by excluding the entities and get new Macro-F value (81.72) and micro F1 value (84.42). As for Wiki80 dataset, this approach doesnt allow disrelated entities to generate negative samples. By eliminating entity factors, the performance gets significant improvements. Compared with baselines in these conditions, we demonstrate our framework works well when entities are prepared.

4.4 Ablation Study

Because we predict relation types with different information level vectors, we delete or modify some parts of the concatenation vector C to compare the performance with before. We set the experiment that implements on SciERC dataset with e=5 and without order loss as the benchmark for this subsection. Table 4 illustrates all comparisons of EMPNs variation in experiment groups. All the performances are lower than benchmark, illustrating the components in our framework are essential.

i) Delete f_s and f_o that come from word embeddings: Removing the embeddings shows whether prediction relies on them. The performance is 2.4% lower than the benchmark. Embeddings represent original information without other words. Some samples may contain the relation that is not decided by words except entities and cause performance penalties.

ii) Delete hidden states H_i and H_j in entity locations: According to the performance, it lags 20% behind the benchmark. From this observation, we conclude that the hidden states from

Table 2: Comparison with joint models between Our Framework and Baselines

Dataset	Baseline	P	R	F
Scierc	(Eberts & Ulges, 2020)	48.89	39.96	43.23
	(Bekoulis et al., 2018)	45.30	34.45	38.30
	(Zhong & Chen, 2021)	-	-	50.10
	(Luan et al., 2018)	47.60	33.50	39.30
	(Shen et al., 2021)	52.63	52.32	52.44
	(Luan et al., 2019)		-	41.6
	0-1 EMPN(our framework; e=3)	56.84	49.77	52.49
	0-1 EMPN (our framework; e=2; order loss)	51.50	54.51	52.51
Conll2004	(Eberts & Ulges, 2020)	74.11	68.69	71.16
	(Bekoulis et al., 2018)	72.99	63.67	67.01
	(Shen et al., 2021)	73.01	71.63	72.35
	(Adel & Schütze, 2017)	-	-	62.50
	0-1 EMPN(our framework; e=3)	59.56	73.41	64.59
	0-1 EMPN(our framework; e=2; order loss)	60.20	65.34	62.39
	0-1 EMPN(our framework; e=7; excluded)	83.65	81.28	81.72
Wiki80	(Eberts & Ulges, 2020)	33.03	41.86	36.79
	(Bekoulis et al., 2018)	71.34	31.28	43.49
	0-1 EMPN(our framework; e=7)	74.79	74.38	74.02
	0-1 EMPN(our framework; e=7; order loss)	75.15	74.59	74.23

Table 3: Comparison with graph models and sequence models between Our Framework and Baselines

Dataset	Baseline Type	Baseline	P	R	F
Scierc	Graph models	(Zhang et al., 2018)		-	65.3
		(Guo et al., 2019)	69.60	72.60	70.70
	Sequence models	(Veyseh et al., 2020)		-	69.92
		(Zhang et al., 2017)	67.58	63.9	65.06
		(Zhou et al., 2016)	63.96	61.93	61.03
		EMPN(our framework; e=5)	71.19	72.27	71.49
		EMPN (our framework; e=5; order loss)	75.00	71.48	72.80
Conll2004	Graph models	(Guo et al., 2019)	92.42	92.18	92.30
	Sequence models	(Zhou et al., 2016)	71.26	71.70	70.96
		EMPN(our framework; e=2)	93.97	93.81	93.85
		EMPN(our framework; e=3; order loss)	92.98	93.10	93.00
Wiki80	Graph models	(Guo et al., 2019)	61.00	52.70	56.60
	Sequence models	(Zhang et al., 2017)	71.26	71.70	70.96
		(Zhou et al., 2016)	75.07	74.85	74.48
		EMPN(our framework; e=2)	75.96	75.48	75.35
		EMPN(our framework; e=10; order loss)	75.85	75.14	74.83

the encoder contain key factors in prediction. They represent the information gathered from other words to entity mentions.

iii) Delete trigger vectors L and re-encoder outputs U from trigger vectors: In order to study the promotion of EMPN, we design this experiment. It is equivalent to e=0, meaning that there is no trigger. The performance is the best among all experiments, indicating that combination of entity embeddings and hidden states in entity locations is the effective choice when EMPN is absent.

iv) Delete one of trigger vectors L from trigger vectors: We set up this to see whether the trigger vectors are unnecessary. The performance is not up to benchmark, which means trigger vectors are required by our framework.

v) Delete re-encoder outputs U from trigger vectors: Same to (iv), The performance is lower than the benchmark indicates that the selected triggers need to be encoded to generate text level information.

vi) Add attention mechanism to output layer: We design this experiment with the aim of verifying whether the output layer can be set learnable weights for different information level vectors. The performance falls from the benchmark because all vectors are independent. When they are provided with different weights, our framework causes similar problems to the cases above on account of weakened error transfer.

Table 4: Ablation study on the test set of SciERC

Experiment Group	Modified mode	Macro-F	Micro-F
(i)	H_i, H_j, L, U	68.09	78.95
(ii)	f_s, f_o, L, U	49.75	64.68
(iii)	f_s, f_o, H_i, H_j	70.48	81.11
(iv)	f_s, f_o, H_i, H_j, L	69.41	78.85
(v)	f_s, f_o, H_i, H_j, U	69.73	80.29
(vi)	f_s, f_o, H_i, H_j, L, U with output layer attention	70.22	80.70

4.5 BERT Model Study

We replace all encoders in our framework with a BERT model (Devlin et al., 2019) to verify performance. The BERT model encodes entity vectors, origin text tokens, and selected trigger tokens. As for the above subsection, we use the same benchmark. According to table 5, the performance that generates 4-length triggers is better than the 5-length. Compared with (Veyseh et al., 2020), the promotion of ours is limited. BERT model is unable to output the cell state that EMPN needs. As a solution, an extra LSTM encoder follows the BERT model, which causes the degradation in fine-tuning process.

Table 5: BERT Model study on the test set of SciERC

Experiment Group	Modified mode	Macro-F	Micro-F
(i)	BERT model with our framework(e=5)	77.65(e=5)	85.43(e=5)
(ii)	BERT model with our framework(e=4)	78.82(e=4)	85.22(e=4)
(Veyseh et al., 2020)	BERT model with Veysehs work	78.24	-

4.6 Case Study

We output the relation triggers to observe the change of them in different type of relation. The words in trigger are generated step by step so that they are in order. EMPN selects the maximal probability word vectors with head and tail in order to bridge them by word paths. We conduct the analysis with a number of right and wrong cases and show 4 of them in figure 3 and figure 4. We choose the triggers generated in SciERC dataset with e=5 with order loss and exhibit them in the grids.

For the samples in figure 2, they share the same origin text but achieve different triggers. The triggers for the first consist of by, the, using, test and a quotation mark. These words are used to indicate the relation type. The words by using can determine USED-FOR type. The triggers for the second are around two entity mentions and thus illuminate words on both sides of entities that tell the type. These trigger words give the same status for two entities and show the CONJUNCTION type. The last one contains task, classification, viewpoint, which can be interpreted as evaluation mode. All these cases show that EMPN is able to select relation triggers and assists in promoting performance in RE.

For the sample in figure 3, the selected triggers are on, these, the, two, in, which cannot be inferred to EVALUATE-FOR type. The phrase classification tasks in object contributes to this type. When the key factors for relation types are in the entities, the performance relies on the encoder. In this case, the information of tail entity is not fully extracted. Next, the fault propagates to EMPN, and letting it select triggers useless to predict relation types. Furthermore, the triggers in this case are like USED-FOR word mode. When the triggers consist of nouns and prepositions before and between two entities, the relation can be easily judged as USED-FOR.

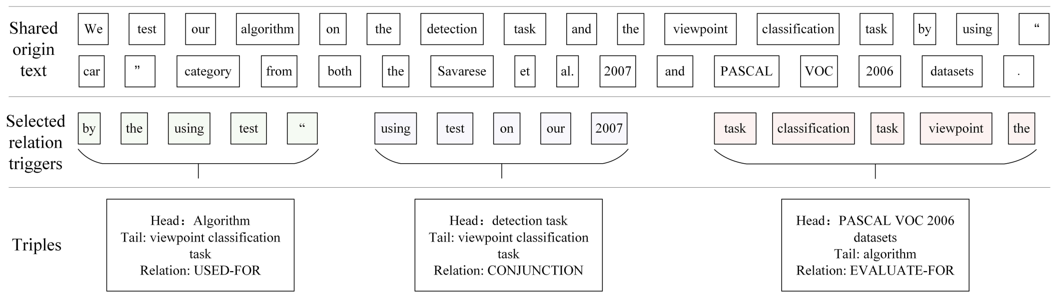

Figure 2: Right cases

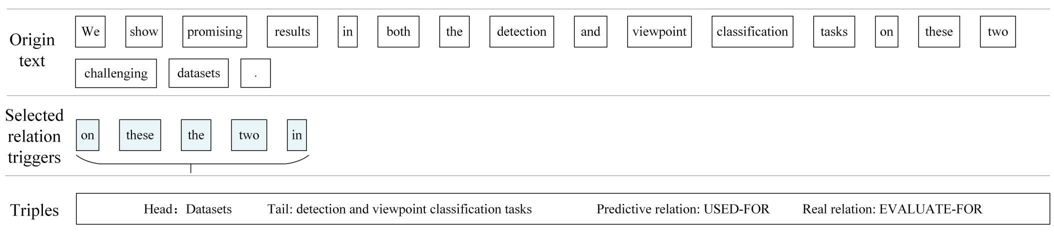

Figure 3: Wrong cases

5 Conclusion

In this paper, we propose a new framework for extracting relation in RE. We employ EMPN as the decoder to generate relation triggers and re-encode them with a new encoder. This framework increases the distinction for different samples with identical origin text. EMPN stops the propagation of interference words in which attention mechanism allows with low weights. We design evaluated mask mechanism for Pointer Network to select words in different locations. This is the pivotal key for verifying the summit performance. We utilize Relative Supervised Learning thought to supervise selected words from relation types. Through extensive experiments on SciERC, Conll 2004, Wiki80 dataset, we demonstrate the effectiveness of our framework, and the validity of the triggers.

In the future, we are determined to study the encoder parts to adjust our framework for both entity extraction and relation extraction. We will also optimize EMPN and enable it to stop generation of its own. Therefore, the length value e will not be the fixed but the maximum.

References

Heike Adel and Hinrich Schütze. Global normalization of convolutional neural networks for joint entity and relation classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1723–1729, 2017.

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. Probing linguistic features of sentence-level representations in neural relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1534–1545, 2020.

Dang Trung Anh and Nguyen Thi Thu Trang. Abstractive text summarization using pointer-generator networks with pre-trained word embedding. In *Proceedings of the Tenth International Symposium on Information and Communication Technology*, pp. 473–478, 2019.

Long Bai, Xiaolong Jin, Chuanzhi Zhuang, and Xueqi Cheng. Entity type enhanced neural model for distantly supervised relation extraction (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 13751–13752, 2020.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, 2019.

Markus Eberts and Adrian Ulges. Span-based joint entity and relation extraction with transformer pre-training. In *ECAI 2020*, pp. 2006–2013. IOS Press, 2020.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1631–1640, 2016.

Zhijiang Guo, Yan Zhang, and Wei Lu. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 241–251, 2019.

Zhijiang Guo, Guoshun Nan, Wei Lu, and Shay B Cohen. Learning latent forests for medical relation extraction. In *IJCAI*, pp. 3651–3657, 2020.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.

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

Halil Kilicoglu, Graciela Rosemblat, Marcelo Fiszman, and Dongwook Shin. Broad-coverage biomedical relation extraction with semrep. *BMC bioinformatics*, 21:1–28, 2020.

Tatsuki Kuribayashi, Paul Reisert, Naoya Inoue, and Kentaro Inui. Examining macro-level argumentative structure features for argumentative relation identification. Technical report, 2017.

Yang Li, Guodong Long, Tao Shen, Tianyi Zhou, Lina Yao, Huan Huo, and Jing Jiang. Self-attention enhanced selective gate with entity-aware embedding for distantly supervised relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8269–8276, 2020.

Yi Liao, Xin Jiang, and Qun Liu. Probabilistically masked language model capable of autoregressive generation in arbitrary word order. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 263–274, 2020.

Shuang Liu, Nannan Tan, Yaqian Ge, and Niko Lukač. Research on automatic question answering of generative knowledge graph based on pointer network. *Information*, 12(3):136, 2021.

Colin Lockard, Prashant Shiralkar, Xin Luna Dong, and Hannaneh Hajishirzi. Zeroshotceres: Zero-shot relation extraction from semi-structured webpages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8105–8117, 2020.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 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*, pp. 3219–3232, 2018.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3036–3046, 2019.

Tapas Nayak and Hwee Tou Ng. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8528–8535, 2020.

Jian Ni, Taesun Moon, Parul Awasthy, and Radu Florian. Cross-lingual relation extraction with transformers. *arXiv preprint arXiv:2010.08652*, 2020.

Yihe Pang, Jie Liu, Lizhen Liu, Zhengtao Yu, and Kai Zhang. A deep neural network model for joint entity and relation extraction. *IEEE Access*, 7:179143–179150, 2019.

Seong Sik Park and Harksoo Kim. Relation extraction among multiple entities using a dual pointer network with a multi-head attention mechanism. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pp. 47–51, 2019.

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

Amir Pouran Ben Veyseh, Thien Nguyen, and Dejing Dou. Improving cross-domain performance for relation extraction via dependency prediction and information flow control. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

Meng Qu, Tianyu Gao, Louis-Pascal Xhonneux, and Jian Tang. Few-shot relation extraction via bayesian meta-learning on relation graphs. In *International Conference on Machine Learning*, pp. 7867–7876. PMLR, 2020.

Benjamin Roth, Costanza Conforti, Nina Poerner, Hinrich Schütze, and Sanjeev Kumar Karn. Neural architectures for open-type relation argument extraction. *Natural Language Engineering*, (2): 219–238, 2018.

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, 2017.

Hamed Shahbazi, Xiaoli Fern, Reza Ghaeini, and Prasad Tadepalli. Relation extraction with explanation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6488–6494, 2020.

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

Fei Sun, Peng Jiang, Hanxiao Sun, Changhua Pei, Wenwu Ou, and Xiaobo Wang. Multi-source pointer network for product title summarization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 7–16, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Amir Pouran Ben Veyseh, Franck Dernoncourt, Dejing Dou, and Thien Huu Nguyen. Exploiting the syntax-model consistency for neural relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8021–8032, 2020.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in Neural Information Processing Systems*, 28:2692–2700, 2015.

Zhenkai Wei, Yu Hong, Bowei Zou, Meng Cheng, and Jianmin Yao. Dont eclipse your arts due to small discrepancies: Boundary repositioning with a pointer network for aspect extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3678–3684, 2020.

Hai Ye and Zhunchen Luo. Deep ranking based cost-sensitive multi-label learning for distant supervision relation extraction. *Information Processing & Management*, 57(6):102096, 2020.

Y. Yin, L. Song, J. Su, J. Zeng, C. Zhou, and J. Luo. Graph-based neural sentence ordering. In *Twenty-Eighth International Joint Conference on Artificial Intelligence IJCAI-19*, 2019.

Yongjing Yin, Fandong Meng, Jinsong Su, Yubin Ge, Lingeng Song, Jie Zhou, and Jiebo Luo. Enhancing pointer network for sentence ordering with pairwise ordering predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 9482–9489, 2020.

Jingli Zhang, Yu Hong, Wenxuan Zhou, Jianmin Yao, and Min Zhang. Interactive learning for joint event and relation extraction. *International Journal of Machine Learning and Cybernetics*, 11(2): 449–461, 2020.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 35–45, 2017.

Yuhao Zhang, Peng Qi, and Christopher D Manning. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2205–2215, 2018.

Zexuan Zhong and Danqi Chen. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 50–61, 2021.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)*, pp. 207–212, 2016.

Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1): 44–53, 2018.

A Appendix

We also explore how the length of generated triggers and order loss affects the performance. Figure 4 describe the trend lines on the three datasets. By observing e, the summit performances of the datasets are different. For SciERC and Wiki80, they are marked as thesis abstract or encyclopedia style and reach most of their summits when e5. For Conll 2004, the dataset about narrative style articles, reaches its summit when e2,3. This demonstrates the relations in thesis abstract need more words as triggers to express while those in the narrative need less. Comparing the zero-one task and normal task, we find the summits move forward in e list because of the no relation samples.

By observing order loss, it promotes the performance of SciERC but degrades some of the others. Long trigger is more sensitive to word order. According to the analysis of parameter e, SciERC has longer trigger than others so that it benefits from order loss. By observing zero-one tasks, it can be ensured that the summit shares the same tendency with EMPN and moves forward in the result sequences with the effect of order loss. The added samples related to no triggers contribute to this, cutting the average length of trigger. These three factors demonstrate that triggers generated by EMPN are valid in RE.

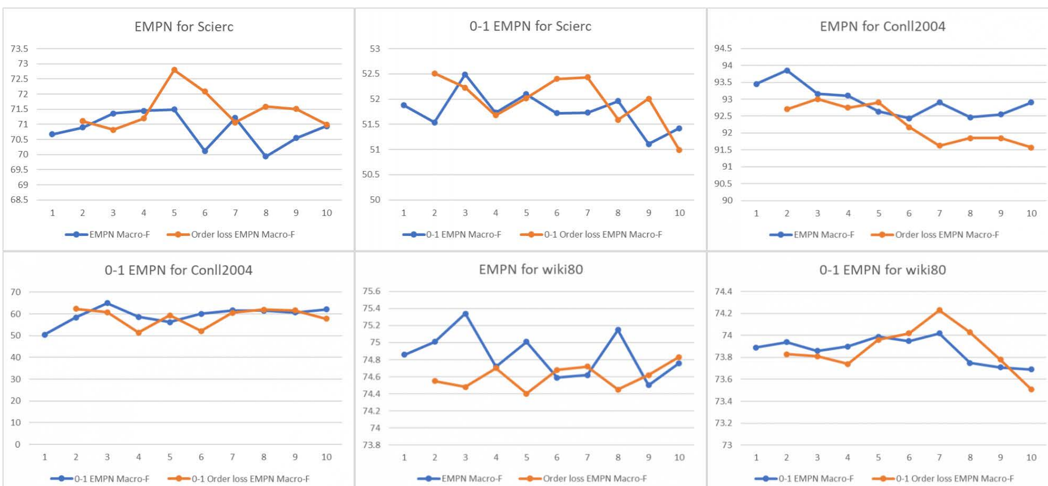

Figure 4: The macro-F trend of generated trigger length