Relation Extraction based on Data Partition and Representation Integration

Jiapeng Zhao[1,2], Panpan Zhang[1,2], Tingwen Liu[1,2]*, Zhenyu Zhang[1,2], Yanzeng Li[1,2], Jinqiao Shi[3]
[1] Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
[3] School of Cyberspace Security, Beijing University of Posts and Telecommunications
{zhaojiapeng, zhangpanpan, liutingwen, zhangzhenyu1996, liyanzeng}@iie.ac.cn, shijinqiao@bupt.edu.cn

***Abstract*—Relation extraction (RE) is the cornerstone of many natural language processing applications. The success of machine learning algorithms generally depends on the embedding representation of data, since it involves different explanatory factors of variation behind the data. This paper explores integrating multiple representations to improve relation extraction. Note that the shortest dependency path (SDP) can retain relevant information and eliminate irrelevant words in the sentence, we partition the input dataset into multiple subsets based on the deformed SDP. For the input dataset and each subset, we train different encoders respectively. The input dataset encoder pays more attention to words in SDP, while the subset encoders pay more attention to words beyond SDP. In this way, we can get multiple embedding representations of diversity for the same sentence and improve the performance of RE by integrating them with predefined strategies. Experimental results on a widely used dataset demonstrate the effectiveness of our approach.**

***Index Terms*—relation extraction, dataset partition, representation integration, shortest dependency path, representation learning**

I. Introduction

Relation extraction, also known as relation classification, involves mining semantic relationships between two entities (subject and object) in a sentence. For instance, given a sample sentence

> "The [earthquake]$_{\text{subject}}$ triggered by the [eruption]$_{\text{object}}$ of Thera struck."

the goal would be automatically mining a "cause-effect" relationship between subject entity mention "earthquake" and object entity mention "eruption". Extracting relations between two entities in a sentence plays an important role in many applications of natural language processing, such as knowledge base population [1], question answering [2] and knowledge discovery [3]. Thus, it has attracted considerable attention from researchers over the past decades [4]–[6] and improving the performance of relation extraction has always been an important issue.

Supported by the National Key Research and Development Program of China (grant 2016YFB0801003), the Strategic Priority Research Program of Chinese Academy of Sciences (grant XDC02040400), the Key Research and Development Program for Guangdong Province (grant No.2019B010137003) and the National Natural Science Foundation of China (grant No.61902394).

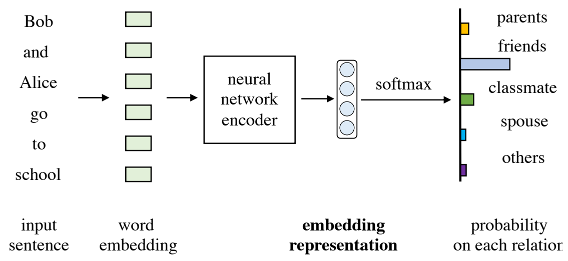

Fig. 1. Illustration of most relation extraction methods.

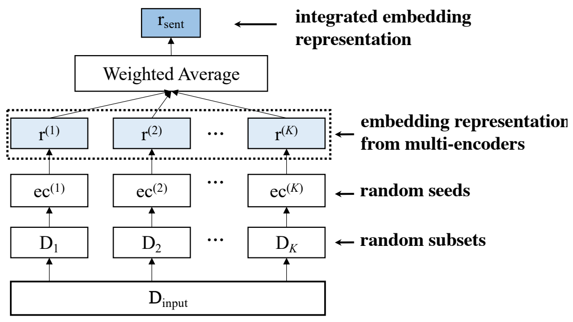

Fig. 2. Random seeds or random subsets methods for learning a better representation. D_{input} is the initial dataset, D_i indicates the i-th subsets of D_{input} for random subsets and the i-th copy of D_{input} for random seeds. $ec^{(i)}$ indicates the i-th encode which is a neural network with arbitrary structure. $r^{(i)}$ indicates representations output from $ec^{(i)}$. $r^{(i)}$ is the i-th representation from the i-th encoder, r_{sent} is the representation after integration.

Recently, it has witnessed many progresses in relation extraction community. These works could be seen as transforming the sentence into an embedding representation with various neural networks and performing classification with the embedding representation, as shown in Figure 1. The performance of relation extraction generally depends on the embedding representation. For most neural network models, learning a good embedding representation is a key factor for the success of relation extraction.

So what's the definition of a good embedding representation? As Yoshua Bengio et al. argued in paper [7]: A good representation is the one that is useful as input to a supervised predictor. In this paper, we focus on learning a better representation based on the existing representation learned by various neural networks.

Learning multiple representations for an instance with random subsets or random seeds methods [8] and integrating these representations is an effective way to find a better embedding representation, as shown in Figure 2. The random subsets method means $\mathcal{D}_1 ... \mathcal{D}_K$ are random subsets of $\mathcal{D}_{input}$. The random seeds method means $\mathcal{D}_1 ... \mathcal{D}_K$ equals $\mathcal{D}_{input}$, but each encoder $ec^{(i)}$ is initialized with different seeds. According to the theory of ensemble learning [8], [9], the diversity of multi-encoders $ec^{(i)}$ and the performance of each encoder are two key factors to boost performance. Both the random subsets and the random seeds methods can encourage the diversity of each encoder $ec^{(i)}$. Although achieving improved results, there are some weeknesses exist in these works.

- **random subsets** The main weakness is that each sentence is assigned to some subsets $\mathcal{D}_i$ randomly. When the number of sentences in each subset is too small, it will reduce the performance of the encoder since neural networks are data-hungry models. Conversely, some subsets may share too many same instances, this will drive down the diversity of each encoder $ec^{(i)}$.
- **random seeds** The main weakness is that since the selection of seeds will directly affect the model performance, but seeds are usually selected randomly without any theoretical basis. At the same time, some subsets will share the same sentences, this will also drive down the diversity between each encoder. Besides, neural network models are hard to explain inherently and the predictions of neural networks are unstable, random subsets or random seeds aggravate the instability and the lack of interpretability.
- **multi-network structures** This means each encoder $ec^{(i)}$ should have different network structures to encourage diversity. The integration of multi-network structures is not conducive to evaluate the optimal performance of a single network structure. In this paper, we only discuss the promotion of representation integration to the same network structure.

To overcome these limitations, we propose a data partition and representation integration (DPRI) method to learn a better representation. We first introduce the shortest depencency path (SDP) to solve the randomness. Natural language is structured data. According to previous studies [10]–[12] on relation extraction, the relationship between two named entities in the same sentence is typically captured by the SDP between the two entities. With this operation, sentences in a same subset may have the same or similarity syntactic structures, while sentences in different subsets may have different syntactic structures. The syntactic structure can reflect the semantic information of a sentence, which saves the diversity between different encoders. In comparison with random subsets methods, since the SDP of each sentence is fixed, the sentence will be assigned to a fixed subset based on it's SDP instead of being assigned randomly. In our setting, each sub encoder $ec^{(i)}$ share the same structure, which allows us to evaluate how much the integrated representation r_{sent} performs better than any source representation $r^{(i)}$ under our DPRI.

In summary, partitioning the dataset reasonably, training the relational encoder and integrating multiple embedding representations are three key factors to improve relation extraction.

For partitioning the dataset, the fundamental goal is prompting encoders of subsets to focus on different words of sentences from the encoder of the input dataset as much as possible. The SDP retains the most relevant information to relation classification while eliminating irrelevant words in the sentence [13]. Since we partition the dataset based on SDP, sentences in the same subset share the similar SDPs. The encoder of each subset will pay more attention to words beyond SDP, while the encoder of the input dataset will pay more attention to words in SDP. It drives each encoder to focus on different information on the first layers of the network. This is quite different from those works [14], [15] which tuned embedding vectors at last layers of the network. Meanwhile, the encoder of each subset needs enough training data to ensure performance. Usually, a small amount of data is hard to learn effective representations for an encoder. In hence, we generalize the SDP by replacing words with their part of speech (named as deformed SDP) and the inclusion relationships between deformed SDP to increase the data amount of each subset.

For learning relational encoders, existing works mainly utilize RNNs to capture a token's contextual information. Aiming at the insufficiency of modeling long-range dependency information, we introduce self attention [16] model to mitigate this problem. The self-attention directly draws the global dependencies of the inputs and conducts direct connections between two arbitrary tokens in a sentence regardless of their distances. Another problem of RNNs is lacking a way to tackle the tree-structure of the inputs. The syntactic structure of a sentence is usually tree-structured. We introduce the graph convolutional to solve the shortcomings of RNNs in tackling the tree-structure of the inputs. In comparison with RNNs, our encoder performs better on capturing long-range dependency information and syntactic structure information than existing works.

For integrating multiple representations, we investigate integrating representations of multiple encoders with a kind of convolutional neural network. We can find an optimum representation from multi representations and their combinations. Through our empirical research, it's better than the weighted average method.

In summary, our contributions are as follows:

- We propose a learning better representation method for relation extraction named DPRI. When a neural network model is given, our DPRI able to produce a better representation than the given one. With our integrated representation, one can get better relation extraction results.
- We propose to partition the dataset based on the SDP of the sentence. It performs better than the random seeds

or random subsets methods in our experiment. DPRI has no randomness and improves the stability of the representation integration methods.

- We test the effectiveness of DPRI on the sequence-based model, dependency-based model and combination model (combination of sequence-based model and dependency-based model). On a widely used relation extraction dataset TACRED, DPRI shows stable effectiveness.

II. Related Work

A. Relation Extraction

Relation extraction approaches could be divided into two types, namely the sequence-based one and the dependency-based one. The sequence-based approaches extract discrete features from the sentence or take sentences as a sequence of tokens directly. They leverage different neural networks to extract relations, including convolutional neural networks [17], recurrent neural networks [18], [19] , the combination of both [20] and transformer [21]. The dependency-based approaches integrate dependency structure and semantic information in the sentence, based on dependency grammar, which can present the implicit semantic information of a full sentence. [22] first split the dependency graph into two DAGs, then extend the tree LSTM model over these two graphs for n-ary relation extraction. Song et al use graph recurrent networks to directly encode the whole dependency graph without breaking it. Various pruning strategies have also been proposed to distill the dependency information in order to further improve the performance. [13] adopts neural models to encode the shortest dependency path. [23] applies the LSTM model over the LCA subtree of two entities. [24] combines the shortest dependency path and the dependency subtree. [25] adopts a shortest dependency path-centric pruning strategy to perform relation extraction. Based our analysis on existing works, PA-LSTMs [19] and CGCN [25] are the two types of most representative and outstanding models. We tested the improvement of DPRI on these two types models.

Another type of works on relation extraction is the pre-trained language models [26] that are trained with a large number of external corpus. Some works [27], [28] further use external knowledge to enhance pre-trained models. Note that the base encoder of our DPRI is replaceable. Replacing the base encoder with those pre-trained language models is also possible.

B. Ensemble Learning

Some deep ensemble learning works have been attempted before. For instance, many ensemble algorithms have been applied to neural networks, such as the Super Learner [29] and AdaBoost [30]. These works ensemble multiple networks on the prediction results while our work on the representation. The behavior of ensemble learning with deep networks is still not well studied and understood [31], [32]. DPRI could be seen as an attempt of ensemble learning on the representation. Note that DPRI also compatible with other ensemble methods.

Other works related to ensemble multiple representations only being explored in image classification. [14] mainly processes on the critical state of data, they trained new layers for the data which are hard to classify with previous layers. [33] adopts multi view learning to learn multiple representations. [15] applies an attention mechanism to encourage diversity of each representation and then integrate multiple representations. However, these's not any related researches in relation extraction till now.

III. Methods and Technical Solutions

In this section, we describe our data partition and representations integration method for relation extraction in detail, as shown in Figure 3. The DPRI consists of three key modules: data partition module, sentence encoder module and representations integration module.

Given a dataset $\mathcal{D}_{input}$, the data partition module partitions $\mathcal{D}_{input}$ into K subsets $\mathcal{D}_{sub} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_K\}$, where $\mathcal{D}_i$ is the i-th subset of $\mathcal{D}_{input}$. Note that these subsets may intersects each other. The sentence encoder module will first train $K+1$ encoders based on $\mathcal{D}_{sub}$ and $\mathcal{D}_{input}$, expressed in $f_{sub} = \{f_1, f_2, ..., f_K\}$ and f_0 respectively. The representations integration module will integrate representations of $K+1$ encoders to calculate the final representation $\mathbf{r}_{\text{sent}}$, which is used for relation extraction.

A. Data Partition Module

The data partition module aims at finding appropriate subsets $\mathcal{D}_{sub}$ from $\mathcal{D}_{input}$. The keys to success are the high performance of individual encoders and diversity among multiple representations [15], [34]. In hence, the diversity of $\mathcal{D}_{sub}$ and large data amount to train each encoder are encouraged.

As described above, the relationship between two named entities in the same sentence is typically captured by the SDP between the two entities [10]–[13]. When partitioning dataset based on SDP, each subset in $\mathcal{D}_{sub}$ involves distinct key information. With this operation, sentences in a same subset share same or similar SDP, while sentences in different subsets have different SDPs. It helps the diversity between different subsets in comparison with random subsets.

In common sense, the deep neural network requires large amounts of training data to ensure performance. Partitioning $\mathcal{D}_{input}$ directly based on the SDP may result in insufficient data amount of $\mathcal{D}_{sub}$. So we further replace words with it's parts of speech (named deformed-SDP for convenience description), and then perform substring matching for different deformed-SDPs. If sentence A's deformed-SDP is a substring of sentence B's deformed-SDP, B is assigned to the subset which A belongs to. Note that a sentence may belongs to multiple data subsets. Finally, we reserve the top K subsets.

B. Sentence Encoder Module

We named our sentence encoder module as CGCN+, since it could be seen as an improvement of CGCN [25]. Given a sentence, denoted as $\mathcal{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$, n is the number of words in the sentence, $\mathbf{x}_i \in \mathbb{R}^{d_w}$ is the representation of the

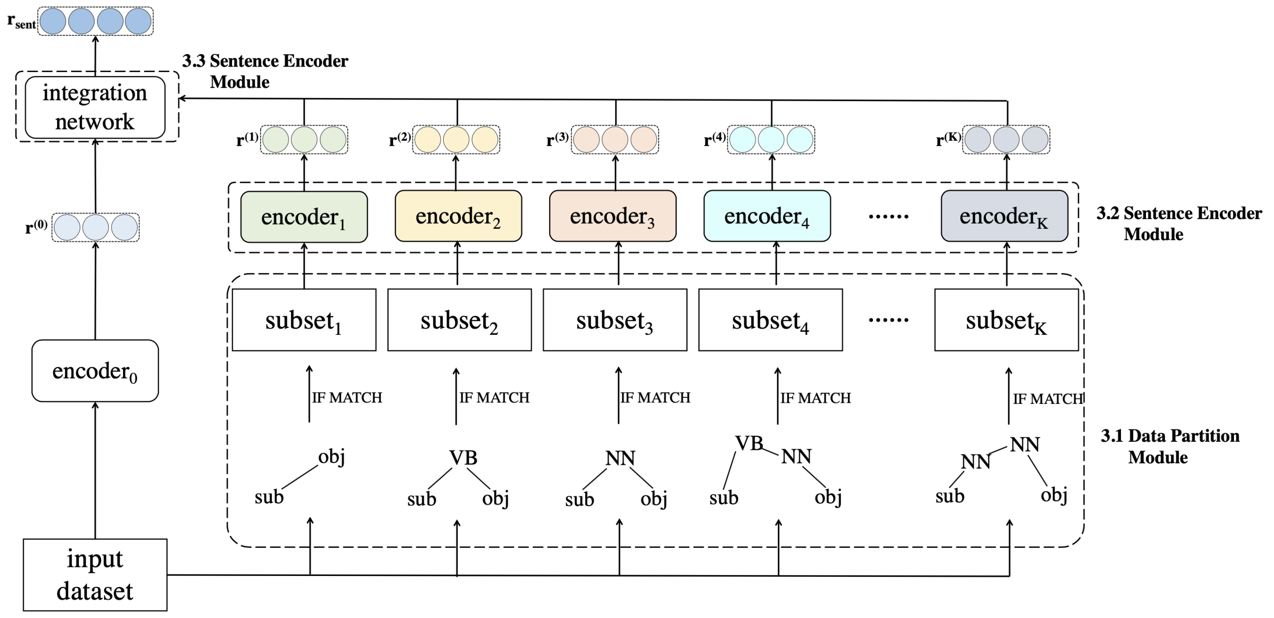

Fig. 3. The workflow of our DPRI. In the training stage, all sentences in the input dataset will be fed into encoder$_0$, other encoders only receive sentences which match corresponding SDP. Each encoder is trained independently. They share the same network structure but learn different parameters to help each encoder output different representations for a same sentence. In the testing stage, only the sentence has the corresponding SDP, the encoder will output an embedding representation, otherwise it will output a zero vector. The integration network will find a better representation from multi-representations and their combinations for the final prediction.

i-th word. Figure 4 summarized the architecture of CGCN+. The objective of encoder is to learn a high-dimensional non-linear function f_j which maps $\mathcal{X}$ to a feature space $\mathbb{R}^{d_r}$. The network architecture of f_j is detailed in the following section. For each word $\mathbf{x}_i$ in $\mathcal{X}$, it doesn't contain the sequential order of input or any contextual information which could be used for disambiguation. To solve this problem, $\mathcal{X}$ is fed into a bi-direction long short-term memory (BiLSTM) network.

$$\mathbf{H}_{\text{bilstm}} = \text{BiLSTM}(\mathcal{X}) \tag{1}$$

Since the BiLSTM faces long-range dependencies problem between tokens, we adopt self-attention that directly capture the relationships between two tokens regardless of their distance. It adopts a multi-head attention mechanism [16] that only requires a single sequence to compute its representation.

$$\mathbf{H}_{\text{mha}} = \text{MultiHeadAttention}(\mathbf{H}_{\text{bilstm}}) \tag{2}$$

We further use the graph convolutional [35] to capture the syntactic dependency between words in a sentence. We follow the usage of graph convolution neural in previous work [25] to model trees by converting each tree into its corresponding adjacency matrix $\mathbf{A}$, where $A_{ij} = 1$ if there is a dependency edge between tokens i and j.

$$\mathbf{H}_{\text{gcn}} = \text{GCN}(\mathbf{H}_{\text{mha}}, \mathbf{A}) \tag{3}$$

Only applying the graph convolution operation may lead to node representations with drastically different magnitudes, since the degree of a token varies a lot. This will bias the sentence representation towards favoring high-degree nodes regardless of the information carried in the node. Furthermore, the information of a node will never carried over to itself, since nodes never connect to themselves in a dependency tree. These

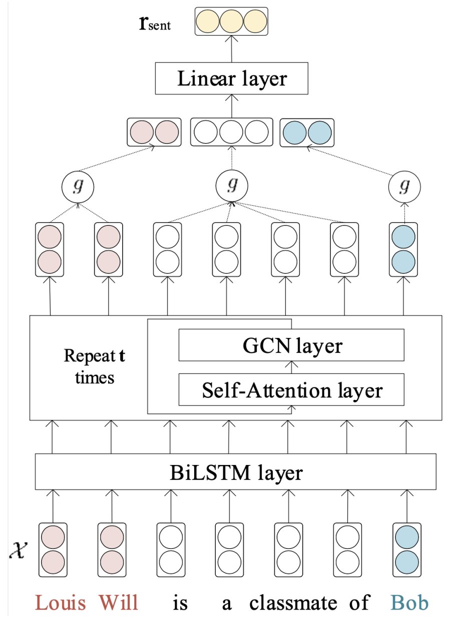

Fig. 4. The architecture of encoder.

issues were resolved by normalizing the activations in the graph convolution before feeding it through the nonlinearity and adding self-loops to each node.

The output of GCN will be fed into a new multi-head attention layer. After applying an t-layers self-attention stacked with GCN over hidden representations of each token, the hidden representation $\mathbf{H}_{\text{gcn}}^{(t)}$ involves contextual information, long range dependency information and syntactic information. The sentence representation $\mathbf{h}_{sent}$ is as follows:

$$\mathbf{h}_{\text{sent}} = g(\mathbf{H}_{\text{gcn}}^{(t)}) \tag{4}$$

$g : \mathbb{R}^{d\times n} \rightarrow \mathbb{R}^{d}$ is a max pooling function that maps from n output vectors to the sentence vector. Since information close to entity tokens in the dependency tree is often central to relation classification. Therefore, the subject representation $\mathbf{h}_s$ from $\mathbf{H}^{(t)}$ as follows: $\mathbf{h}_s = g(\mathbf{H}_{s_1:s_2}^{(t)})$, s_1 indicates the start position and s_2 indicates the end position of the subject in the sentence. The object representation $\mathbf{h}_o$ is calculated similarly. The sentence representation is concatenating representations of the sentence and two entities, and feeding them through a feed-forward neural network φ:

$$\mathbf{r}_{\text{e}} = \varphi([\mathbf{h}_s; \mathbf{h}_{sent}; \mathbf{h}_o]) \tag{5}$$

We use $\mathbf{r}_{\text{e}}^{(j)}$ to indicate the output of j-th encoder, $\mathbf{0}$ is a zero vector which dimension equals to $\mathbf{r}_{\text{e}}$.

$$\mathbf{r}_{\text{e}}^{(j)} = \begin{cases} f_j(\mathcal{X}) & for\ \mathcal{X}\ in\ \mathcal{D}_j\ (0 <= j <= K) \\ \mathbf{0} & for\ \mathcal{X}\ not\ in\ \mathcal{D}_j\ (0 <= j <= K) \end{cases} \tag{6}$$

C. Representations Integration Module

Since each sub-encoder only being trained on corresponding subset, it is only able to output a meaningful representation for the sentence which matches the corresponding SDP, else the representation will be a zero vector with the same dimension. Simply concatenating multiple representations will lead to inferior performance. In hence, the representation integration module aims at learning to integrate multiple representations which involve diversity information to the final representation. It takes multiple representations of a sentence as input. Since we expect finding an optimal embedding representation from single embedding or multi-embedding representation combinations, a convolutional layer is deployed to compute the final representation of the sentence. The convolutional layer first produces local feature vectors around each representation of a sentence. Then, it combines these local feature vectors using a max operation to create a final representation.

The convolutional layer applies a matrix-vector operation to each window of size k of successive windows in $\mathbf{r}_{\text{e}}$. We define the vector $\mathbf{z}_i \in \mathbb{R}^{d_r\times k}$ as the concatenation of a sequence of k representations, d^r is the dimension of $\mathbf{r}_{\text{e}}^{(i)}$:

$$\mathbf{z}_i = (\mathbf{r}_{\text{e}}^{(i-(k-1)/2)}, ..., \mathbf{r}_{\text{e}}^{(i+(k-1)/2)})^{\text{T}} \tag{7}$$

The convolutional layer computes the j-th element of the vector $\mathbf{r}_x \in \mathbb{R}^{d^c}$ as follows, d^c is the number of convolutional units:

$$[\mathbf{r}_{\text{sent}}]_j = \max_{1<i<N}[\sigma(W\mathbf{z}_i + b)]_j \tag{8}$$

Matrix W and vector b are parameters to be learned. ReLU is the rectifier linear unit activation function. The size of the context window k are hyper parameters. A max pooling layer is deployed to capture the significant features produced by the activation layer. The final $\mathbf{r}_{\text{sent}}$ is a feature vector which involves the most important information of multiple representations. We formulate this module as follows:

$$\mathbf{r}_{\text{sent}} = \vartheta(\mathbf{r}_{\text{e}}) \tag{9}$$

Then we fed $\mathbf{r}_{\text{sent}}$ into a linear layer followed by a softmax operation to obtain a probability distribution over relations.

D. Training Under DPRI

Training DPRI in an end-to-end mode will lead to inferior performance. As described in paper [8], end-to-end training will eliminate gradient diversity of each sub encoder. We can simplify the convolution process to a generic averaging layer,

$$\mu(x_1, ..., x_N) = \frac{1}{N}\sum_{n=1}^{N} x_i, \tag{10}$$

that ultimately contributes to some loss ℓ, and consider the derivative of ℓ with respect to some x_i.

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \mu}\frac{\partial \mu}{\partial x_i} = \frac{\partial \ell}{\partial \mu}\frac{1}{N} \tag{11}$$

Notice that gradients back propagated into each encoder is identical and it doesn't depend on x_i. We think that the diversity of the model is reduced with end to end training model.

In hence, the training process contains two stages. The i-th encoder f_i is first trained on the dataset $\mathcal{D}_i = \{(x_1, y_1), ..., (x_{N_i}, y_{N_i})\}$, $0 \le i \le K$, K is the number of subsets, N_i is the labeled instance number of i-th subset. The output of f_i is fed into a softmax function $\theta_i : x \rightarrow P$, P is the probability distribution over some dataset of labels and f_0 indicates the encoder trained by $\mathcal{D}_{global}$. The optimization objective for each encoder f_i is minimize the loss function:

$$\mathcal{L}_{\text{encoder}}(\mathcal{D}_i) = \frac{1}{|\mathcal{D}_i|}\sum_{n=1}^{N_i} \ell(\theta_i(f_i(x_n)), y_n) \tag{12}$$

where ℓ is a cross-entropy loss. Each encoder f_i share the same architecture, but their parameters are different. Since the network architecture and parameters are two mainly key factors to the diversity of the sentence representations. With this setting, on one hand, the effect of data partitioning could be quantification more accurately. On the other hand, we are able to utilize ensemble multiple systems methods to integrate multiple architecture networks.

After the first stage, all parameters of sub-encoder are fixed. The integration network takes $\mathbf{r}_{\text{e}}$ as input. The optimization objective is to minimize the function $\mathcal{L}_{\text{total}}$:

$$\mathcal{L}_{\text{total}}(\mathcal{D}_0) = \frac{1}{|\mathcal{D}_0|}\sum_{n=1}^{N_0} \ell(\theta_{\text{total}}(\vartheta(\mathbf{r}_{\text{e,n}})), y_n) \tag{13}$$

ϑ is the integration network detailed in III-C, $\mathbf{r}_{\text{e,n}}$ indicates multiple learned representants of the n-th sentence, $\theta_{\text{total}} : \mathbf{r}_{\text{e}} \rightarrow P$ is a softmax function.

IV. Empirical Evaluation

A. Dataset

We conduct experiments on an opened dataset on relation extraction **TACRED**. It contains over 106k mention pairs drawn from the yearly TAC KBP[1] challenge. It represents 41 relation types and a special no relation class when the mention pair does not have a relation between them within these categories. Mentions in TACRED are typed, with subjects categorized into person and organization, and objects into 16 fine-grained types (e.g., date and location). This dataset was widely used in evaluating the performance of relation extraction in recent studies [12], [19], [25], [27], [36].

B. Experimental Settings

1) Baseline Methods:

- **PA-LSTMs** [19] is a typical sequence-based method. It employs a position-aware attention mechanism over LSTM outputs, and outperforms several sequence-based models like CNN and PCNN [37].
- **CGCN** [25] is a typical dependency-based method. It first utilize the dependency relation of the sentence to build a dependency graph, and applies a combination of pruning strategy and graph convolutions over the dependency graph. It outperforms many dependency-based models, such as SDP-LSTM [13].
- **Other improved methods**. **PCNN** [37]: It replaces the max-pooling operation of CNN with piece-wise max-pooling and achieves improved results. **SDP-LSTM** [13]: It applies a neural sequence model iteratively along the shortest dependency path between target entities. **SA-LSTM** [36] applies a segment attention layer to employ on the top of the LSTM. The motivation is based on the observation that information pertinent to relations is usually contained within segments (continuous words in a sentence). **AGGCN** [12] utilizes attention to convert rule based hard-pruning strategies of **CGCN** to a soft-pruning one, which automatically learns how to selectively attend to the relevant sub-structures useful for the relation extraction task.

2) Implementation Detail: For our DPRI, the sentence encoder module is replaceable, as described in section III-B. In our experiment setting, PA-LSTMs, CGCN and our own encoder are utilized as the sentence encoder module respectively. With this setting, on one hand, it is to demonstrate that DPRI outperforms other improvement methods; on the other hand, it is to demonstrate the generalization of DPRI.

The global encoder and sub-encoders in a DPRI share the same hyper-parameters. We first use the pre-trained GloVe vectors [38] to initialize the word embedding. The word embedding size is 300. The word that occurs less than 2 times in the training set are replaced by a special 'UNK' token. The embedding vector size for the named entity tags is 30 and an embedding vector of the same size for the part-of-speech tags to concatenate to the word embedding. The final word embedding size of each word is 360. They are fed into the encoder f_{sub} and f_0. The number of BiLSTM layer is set to 1. The self-attention layer and GCN layer repeat 6 times in our setting. The output size of each encoder is set to 200. The representation integration module is a convolutional neural network which contains three types filters with shape 3, 4 and 5. The number of each type filter is set to 200, 600 in total. The dropout of representation integration module is set to 0.5. The output dim of the representation after integration is set to 200. The batch size is set to 50, the number of epochs for DPRI training and sentence encoder training are less than 100. When using PA-LSTMs and CGCN as the sentence encoder module, we follow the hyper-parameter settings in their papers [19], [25], which could also be visited from websites [2] [3].

[1]https://tac.nist.gov/2017/KBP/index.html

TABLE I
Comparison with different models using DPRI or not on TACRED dataset.

Methods	Precision(%)	Recall(%)	F1-value(%)
PA-LSTMs	67.7	63.2	65.4
DPRI (PA-LSTMs)	70.4	64.2	67.2 **(+1.8)**
CGCN	69.9	63.3	66.4
DPRI (CGCN)	71.8	65.8	68.6 **(+2.2)**
CGCN+	66.9	67.5	67.2
DPRI (CGCN+)	71.5	66.2	68.8 **(+1.6)**

C. Experimental Result Analysis

Table I shows the test results on TACRED of DRPI with different sentence encoders. PA-LSTMs, CGCN and CGCN+ indicate that we use these models as independent relation classifiers. DPRI (PA-LSTMs), DPRI (CGCN) and DPRI (CGCN+) indicate that we utilize these models as the sentence encoder module, which is a sub module of the DPRI. The subsets number K is selected according to their performance on the validation set. The best K is 8 for DPRI (PA-LSTM), 7 for DPRI (CGCN) and 13 for DPRI (CGCN+). The CGCN+ outperforms CGCN by 0.8%, this indicates self attention is encouraged to introduce. In Table I, it shows that the F1-value of PA-LSTM, CGCN and CGCN+ are increased by 1.8%, 2.2% and 1.6% respectively when using DPRI. The comparative results tell that DPRI outperforms using model as independent relation classifier.

The reason is that each sub-encoder focus on different words with the encoder of the input dataset. Then the representation from each sub-encoder will involve different explanatory information for a same sentence. The integration module helps selecting a better representation from multiple representations and their combination. In hence, the selected representation most probably performs better than the representation of the encoder only trained from the input dataset.

[2]https://github.com/qipeng/gcn-over-pruned-trees
[3]https://github.com/yuhaozhang/tacred-relation

TABLE II
COMPARISON WITH RANDOM SEEDS AND RANDOM SUBSETS.DP SHORT FOR DATA PERCENT.

Methods	DP(%)	Precision(%)	Recall(%)	F1-value(%)
r-seeds [8]	-	70.0	65.3	67.7
r-subsets [8]	20	71.7	63.5	67.4
	50	71.3	64.3	67.6
	80	71.1	64.7	67.7
DPRI	-	71.5	66.2	**68.8**

TABLE III
COMPARISON WITH OTHER INTEGRATION METHOD.

Methods	Precision(%)	Recall(%)	F1-value(%)
DPRI-common	67.9	66.1	66.9
DPRI-CNN-avg	69.5	66.7	68.1
DPRI-CNN-max	71.5	66.2	**68.8**

Effect of deformed-SDP In order to demonstrate the effectiveness of the data partition module, we also compare our method with random subsets. We set the number of subsets to 5 for random-subsets method since the data amount used for training is nearly close with our DPRI. As shown in Table II, the data percent of each subset is set to 20 , 50 and 80 respectively. We also compare our method with random seeds, which means each sub-encoder is trained with different seeds for the input dataset. We set 5 different seeds corresponding to 5 different representations. When partitioning dataset with deformed-SDP, the sentences in the same subset share the same or similar SDP. The encoder of subsets will pay more attention to the words beyond SDP while the encoder of the input dataset pay more attention to words on SDP. The encoder of subsets is a good supplement to the encoder of the input dataset. Besides, when using random methods, it's uncertain which subset to partition for an instance. This increases the instability of the method in comparison with partitioning dataset based on the deformed-SDP.

Effect of different integration methods We finally compare our representation integration module with other different integration methods. The experimental result is finally presented in Table III. DPRI-common concatenates the representation of the input dataset with representations of subset, and adopts two layers fully connected neural network to integrate these representations. DPRI-CNN stacks all representations and adopt convolutional neurl network to integrate these representations. We have tried different pooling methods like max-pooling (DPRI-CNN-max) and average-pooling (DPRI-CNN-avg), and found that max-pooling outperforms other pooling method. The DPRI-CNN-max outperforms other integration methods is mainly due to the CNN contains multiple convolutional kernel function. Each convolutional kernel could be seen as a kind of integrating multiple representations to one representation. The max-pooling operation is selecting a best representation from multiple integrated representations. DPRI-common and DPRI-CNN-avg lead to inferior performance. Since if a sentence is not in some subset, the corresponding representation will be set to a zero vector. Intuitively, selecting a best representation will be better than averaging them.

TABLE IV
COMPARISON WITH OTHER IMPROVED METHODS ON TACRED DATASET.

Methods	Precison(%)	Recall(%)	F1-value(%)
PCNN [37]	67.4	57.3	62.0
SDP-LSTM [13]	66.3	52.7	58.7
PA-LSTMs	67.7	63.2	65.4
CGCN	69.9	63.3	66.4
SA-LSTM [36]	69.0	66.2	67.6
AGGCN [12]	73.1	64.2	68.2 (67.5)
DPRI	71.5	66.2	**68.8**

Comparision with improved methods based on GCN and PA-LSTMs From the experiment results in Table IV, our DPRI outperforms other improved methods based on GCN and PA-LSTMs. Table IV presents the comparison results of DPRI with other relation extraction methods. It shows that DPRI outperforms other improved methods like SA-LSTM and AGGCN. The reason that we didn't utilize those improved models as the sentence encoder module are as follows: Although AGGCN is an improvement of GCN, the performance is not as good as they reported in paper [12]. The best F1-value only achieves 67.5% according to their realized codes and hyper-parameters[4].

Comparision with other improved methods Our work also outperforms some pre-trained language models [**?**] 68.2% in F1-value that are trained with a large number of external corpus. Although our method not as good as some of the latest pre-training models [28] 70.6% in F1-value, note that the base encoder of our DPRI is replaceable. Replacing the base encoder with the pre-trained language models is a very interesting direction. It is also the problem that we will study in the future.

V. CONCLUSION

In this paper, we explore the feasibility to partition the dataset to multiple subsets, learning multiple representations from subsets to optimize the representation of the input dataset. We demonstrate the effectiveness of our approach on a widely used relation extraction datasets TACRED. A novel encoder is designed to encode a sentence to an embedding representation. We further prove that partitioning the input dataset based on the SDP is better than the random methods. We test our approach on three encoders, sequence-based encoder, dependency-based encoder and the our improved encoders. To these three encoders, our approach can improve their performance. We also designed a CNN-based integration method, it outperforms those widely used methods of integrating multiple representations.

REFERENCES

[1] B. Shi and T. Weninger, "Open-world knowledge graph completion," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[4]https://github.com/Cartus/AGGCN

[2] A. Abujabal, R. Saha Roy, M. Yahya, and G. Weikum, "Never-ending learning for open-domain question answering over knowledge bases," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 1053–1062.

[3] C. Zhang and J. Han, "Multidimensional mining of massive text data," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 11, no. 2, pp. 1–198, 2019.

[4] D. Zelenko, C. Aone, and A. Richardella, "Kernel methods for relation extraction," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1083–1106, 2003.

[5] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin, "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*, 2013, pp. 74–84.

[6] K. Gábor, D. Buscaldi, A.-K. Schumann, B. QasemiZadeh, H. Zargayouna, and T. Charnois, "Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers," in *Proceedings of The 12th International Workshop on Semantic Evaluation*, 2018, pp. 679–688.

[7] Y. BENGIO, A. COURVILLE, and P. VINCENT, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[8] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra, "Why m heads are better than one: Training a diverse ensemble of deep networks," *arXiv preprint arXiv:1511.06314*, 2015.

[9] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[10] R. C. Bunescu and R. J. Mooney, "A shortest path dependency kernel for relation extraction," in *Proceedings of the conference on human language technology and empirical methods in natural language processing*. Association for Computational Linguistics, 2005, pp. 724–731.

[11] S. He, Z. Li, H. Zhao, and H. Bai, "Syntax for semantic role labeling, to be, or not to be," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 2061–2071.

[12] Z. Guo, Y. Zhang, and W. Lu, "Attention guided graph convolutional networks for relation extraction," *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[13] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin, "Classifying relations via long short term memory networks along shortest dependency paths," in *proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1785–1794.

[14] V. N. Murthy, V. Singh, T. Chen, R. Manmatha, and D. Comaniciu, "Deep decision network for multi-class image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2240–2248.

[15] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon, "Attention-based ensemble for deep metric learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 736–751.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[17] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao *et al.*, "Relation classification via convolutional deep neural network," 2014.

[18] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. 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)*, 2016, pp. 207–212.

[19] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 35–45.

[20] N. T. Vu, H. Adel, P. Gupta *et al.*, "Combining recurrent and convolutional neural networks for relation classification," in *Proceedings of NAACL-HLT*, 2016, pp. 534–539.

[21] P. Verga, E. Strubell, and A. McCallum, "Simultaneously self-attending to all mentions for full-abstract biological relation extraction," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 872–884.

[22] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence n-ary relation extraction with graph lstms," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 101–115, 2017.

[23] M. Miwa and M. Bansal, "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)*, 2016, pp. 1105–1116.

[24] Y. Liu, F. Wei, S. Li, H. Ji, M. Zhou, and W. Houfeng, "A dependency-based neural network for relation classification," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2015, pp. 285–290.

[25] Y. Zhang, P. Qi, and C. D. Manning, "Graph convolution over pruned dependency trees improves relation extraction," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2205–2215.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[27] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "ERNIE: Enhanced language representation with informative entities," in *Proceedings of ACL 2019*, 2019.

[28] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020.

[29] M. J. Van der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Statistical applications in genetics and molecular biology*, vol. 6, no. 1, 2007.

[30] H. Schwenk and Y. Bengio, "Boosting neural networks," *Neural computation*, vol. 12, no. 8, pp. 1869–1887, 2000.

[31] C. Ju, A. Bibaut, and M. van der Laan, "The relative performance of ensemble methods with deep convolutional neural networks for image classification," *Journal of Applied Statistics*, vol. 45, no. 15, pp. 2800–2818, 2018.

[32] S. Tao, "Deep neural network ensembles," *arXiv preprint arXiv:1904.05488*, 2019.

[33] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, and J. Han, "An attention-based collaboration framework for multi-view network representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1767–1776.

[34] M. Opitz, G. Waltner, H. Possegger, and H. Bischof, "Bier-boosting independent embeddings robustly," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5189–5198.

[35] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[36] B. Yu, Z. Zhang, T. Liu, B. Wang, S. Li, and Q. Li, "Beyond word attention: Using segment attention in neural relation extraction," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 5401–5407.

[37] D. Zeng, K. Liu, Y. Chen, and J. Zhao, "Distant supervision for relation extraction via piecewise convolutional neural networks," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1753–1762.

[38] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.