A Novel Global Feature-Oriented Relational Triple Extraction Model based on Table Filling

Feiliang Ren[†,*], **Longhui Zhang**[†], **Shujuan Yin, Xiaofeng Zhao, Shilei Liu**
Bochao Li, Yaduo Liu
School of Computer Science and Engineering
Key Laboratory of Medical Image Computing of Ministry of Education
Northeastern University, Shenyang, 110169, China
renfeiliang@cse.neu.edu.cn

Abstract

Table filling based relational triple extraction methods are attracting growing research interests due to their promising performance and their abilities on extracting triples from complex sentences. However, this kind of methods are far from their full potential because most of them only focus on using local features but ignore the global associations of relations and of token pairs, which increases the possibility of overlooking some important information during triple extraction. To overcome this deficiency, we propose a global feature-oriented triple extraction model that makes full use of the mentioned two kinds of global associations. Specifically, we first generate a table feature for each relation. Then two kinds of global associations are mined from the generated table features. Next, the mined global associations are integrated into the table feature of each relation. This "*generate-mine-integrate*" process is performed multiple times so that the table feature of each relation is refined step by step. Finally, each relation's table is filled based on its refined table feature, and all triples linked to this relation are extracted based on its filled table. We evaluate the proposed model on three benchmark datasets. Experimental results show our model is effective and it achieves state-of-the-art results on all of these datasets. The source code of our work is available at: https://github.com/neukg/GRTE.

1 Introduction

Relational triple extraction (RTE) aims to extract triples from unstructured text (often sentences), and is a fundamental task in information extraction. These triples have the form of (*subject, relation, object*) , where both *subject* and *object* are entities and they are semantically linked by *relation*. RTE is important for many downstream applications.

Nowadays, the dominant methods for RTE are the joint extraction methods that extract entities and relations simultaneously in an end-to-end way. Some latest joint extraction methods (Yu et al., 2019; Yuan et al., 2020; Zeng et al., 2020; Wei et al., 2020; Wang et al., 2020; Sun et al., 2021) have shown their strong extraction abilities on diverse benchmark datasets, especially the abilities of extracting triples from complex sentences that contain overlapping or multiple triples.

Among these existing joint extraction methods, a kind of table filling based methods (Wang et al., 2020; Zhang et al., 2017; Miwa and Bansal, 2016; Gupta et al., 2016) are attracting growing research attention. These methods usually maintain a table for each relation, and each item in such a table is used to indicate whether a token pair possess the corresponding relation or not. Thus the key of these methods is to fill the relation tables accurately, then the triples can be extracted based on the filled tables. However, existing methods fill relation tables mainly based on local features that are extracted from either a single token pair (Wang et al., 2020) or the filled history of some limited token pairs (Zhang et al., 2017), but ignore following two kinds of valuable global features: the global associations of token pairs and of relations.

These two kinds of global features can reveal the differences and connections among relations and among token pairs. Thus they are helpful to both the precision by verifying the extracted triples from multiple perspectives, and the recall by deducing new triples. For example, given a sentence "*Edward Thomas and John are from New York City, USA.*", when looking it from a global view, we can easily find following two useful facts. First, the triple (*Edward Thomas, live_in, New York*) is helpful for extracting the triple (*John, live_in, USA*), and vice versa. This is because the properties of their (*subject, object*) pairs are highly similar: (i) the types of both subjects are same (both are *per-*

[†]Both authors contribute equally to this work and share co-first authorship.
[*]Corresponding author.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 2646–2656
November 7–11, 2021.

sons); (ii) the types of both objects are same too (both are *locations*). Thus these two entity pairs are highly possible to possess the same kind of relation. Second, the mentioned two triples are helpful for deducing a new triple (*New York, located_in, USA*). This is because that: (i) *located_in* requires both its subjects and objects be *locations*; (ii) *located_in* is semantically related to *live_in*; (iii) *live_in* indicates its objects are *locations*. Thus there is a clear inference path from these two known triples to the new triple. Obviously, these global features are impossible to be contained in local features.

Inspired by above analyses, we propose a global feature-oriented table filling based RTE model that fill relation tables mainly based on above two kinds of global associations. In our model, we first generate a table feature for each relation. Then all relations' table features are integrated into a subject-related global feature and an object-related global feature, based on which two kinds of global associations are mined with a *Transformer*-based method. Next, these two kinds of mined global associations are used to refine the table features. These steps are performed multiple times so that the table features are refined gradually. Finally, each table is filled based on its refined feature, and all triples are extracted based on the filled tables.

We evaluate the proposed model on three benchmark datasets: NYT29, NYT24, and WebNLG. Extensive experiments show that it consistently outperforms the existing best models and achieves the state-of-the-art results on all of these datasets.

2 Related Work

Early study (Zelenko et al., 2003; Zhou et al., 2005; Chan and Roth, 2011) often takes a kind of pipeline based methods for RTE, which is to recognize all entities in the input text first and then to predict the relations for all entity pairs. However, these methods have two fatal shortcomings. First, they ignore the correlations between entity recognition and relation prediction. Second, they tend to suffer from the error propagation issue.

To overcome these shortcomings, researchers begin to explore the joint extraction methods that extract entities and relations simultaneously. According to the research lines taken, we roughly classify existing joint methods into three main kinds.

Tagging based methods. This kind of methods (Zheng et al., 2017; Yu et al., 2019; Wei et al., 2020) often first extract the entities by a tagging based method, then predict relations. In these models, binary tagging sequences are often used to determine the start and end positions of entities, sometimes to determine the relations between two entities too.

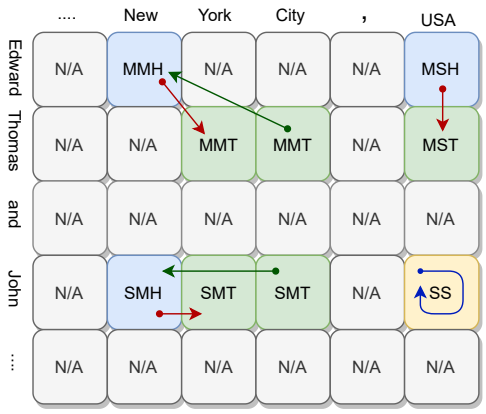

Figure 1: Examples of table filling and decoding strategy. Arrows with different colors correspond to different search routes defined in *Algorithm* 1.

Seq2Seq based methods. This kinds of methods (Zeng et al., 2018, 2019, 2020; Nayak and Ng, 2020) often view a triple as a token sequence, and convert the extraction task into a *generation* task that generates a triple in some orders, such as first generates a relation, then generates entities, etc.

Table filling based methods. This kind of methods (Miwa and Bansal, 2016; Gupta et al., 2016; Zhang et al., 2017; Wang et al., 2020) would maintain a table for each relation, and the items in this table usually denotes the start and end positions of two entities (or even the types of these entities) that possess this specific relation. Accordingly, the RTE task is converted into the task of filling these tables accurately and effectively.

Besides, researchers also explore other kinds of methods. For example, Bekoulis et al. (2018) formulate the RTE task as a multi-head selection problem. Li et al. (2019) cast the RTE task as a multi-turn question answering problem. Fu et al. (2019) use a graph convolutional networks based method and Eberts and Ulges (2019) use a span extraction based method. Sun et al. (2021) propose a multitask learning based RTE model.

3 Methodology

3.1 Table Filling Strategy

Given a sentence $S = w_1 w_2 \dots w_n$, we will maintain a table $table_r$ (the size is $n \times n$) for each relation r ($r \in R$, and R is the relation set). The core

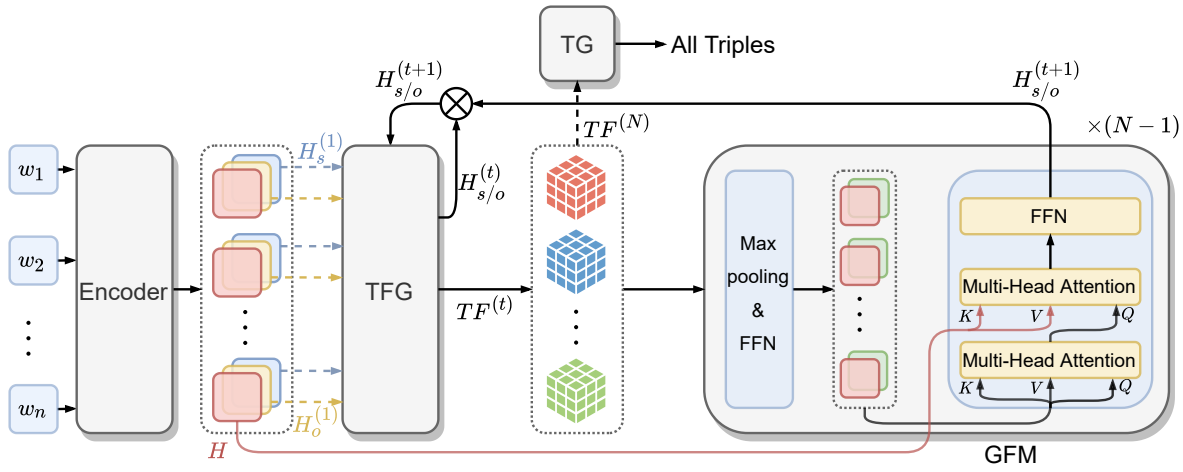

Figure 2: Model Architecture. The dotted arrows to *TFG* means that $H_s^{(1)}$ and $H_o^{(1)}$ will be inputted to *TFG* only at the first iteration. The dotted arrow to *TG* means that $TF^{(N)}$ will be inputted into *TG* only at the last iteration.

of our model is to assign a proper label for each table item (corresponding to a token pair). Here we define the label set as L = {"N/A", "MMH", "MMT", "MSH", "MST", "SMH", "SMT", "SS"}.

For a token pair indexed by the *i*-th row and the *j*-th column, we denote it as (w_i, w_j) and denote its label as l. If $l \in$ {"MMH", "MMT", "MSH", "MST", "SMH", "SMT"}, it means (w_i, w_j) is correlated with a (*subject, object*) pair. In such case, the first character in the label refers to the *subject* is either a multi-token entity ("M") or a single-token entity ("S"), the second character in the label refers to the *object* is either a multi-token entity ("M") or a single-token entity ("S"), and the third character in the label refers to either both w_i and w_j are the head token of the *subject* and *object* ("H") or both are the tail token of the *subject* and *object* ("T"). For example, l = "MMH" means w_i is the head token of a multi-token subject and w_j is the head token of a multi-token object. As for other cases, l = "SS" means (w_i, w_j) is an entity pair; l = "N/A" means w_i and w_j are none of above cases.

Figure 1 demonstrates partial filled results for the *live_in* relation given the sentence "*Edward Thomas and John are from New York City, USA.*", where there are (*subject, object*) pairs of (*Edward Thomas*, *New York City*), (*Edward Thomas*, *New York*), (*Edward Thomas*, *USA*), (*John*, *New York City*), (*John*, *New York*) and (*John*, *USA*).

An main merit of our filling strategy is that each of its label can not only reveal the location information of a token in a subject or an object, but also can reveal whether a subject (or an object) is a single token entity or multi token entity. Thus, the total number of items to be filled under our filling strategy is generally small since the information carried by each label increases. For example, given a sentence $S = w_1 w_2 \dots w_n$ and a relation set R, the number of items to be filled under our filling strategy is $n^2|R|$, while this number is $(2|R|+1)\frac{n^2+n}{2}$ under the filling strategy used in *TPLinker* (Wang et al., 2020) (this number is copied from the original paper of *TPLinker* directly). One can easily deduce that $(2|R|+1)\frac{n^2+n}{2} > n^2|R|$.

3.2 Model Details

The architecture of our model is shown in Figure 2. It consists of four main modules: an *Encoder* module, a *Table Feature Generation* (*TFG*) module, a *Global Feature Mining* (*GFM*) module, and a *Triple Generation* (*TG*) module. *TFG* and *GFM* are performed multiple time with an iterative way so that the table features are refined step by step. Finally, TG fills each table based on its corresponding refined table feature and generates all triples based on these filled tables.

Encoder Module Here a pre-trained *BERT-Base* (Cased) model (Devlin et al., 2018) is used as *Encoder*. Given a sentence, this module firstly encodes it into a token representation sequence (denoted as $H \in \mathbb{R}^{n \times d_h}$).

Then H is fed into two separated *Feed-Forward Networks* (*FFN*) to generate the initial *subjects* feature and *objects* feature (denoted as $H_s^{(1)}$ and $H_o^{(1)}$ respectively), as written with Eq. (1).

$$\begin{aligned} H_s^{(1)} &= W_1 H + b_1 \\ H_o^{(1)} &= W_2 H + b_2 \end{aligned} \tag{1}$$

where $W_{1/2} \in \mathbb{R}^{d_h \times d_h}$ are trainable weights and $b_{1/2} \in \mathbb{R}^{d_h}$ are trainable biases.

TFG Module We denote the *subjects* and *objects* features at the t-th iteration as $H_s^{(t)}$ and $H_o^{(t)}$ respectively. Then taking them as input, this module generates a table feature for each relation.

Here the table feature for the relation r at the t-th iteration is denoted as $TF_r^{(t)}$, and it has the same size with $table_r$. Each item in $TF_r^{(t)}$ represents the label feature for a token pair. Specifically, for a pair (w_i, w_j), we denoted its label feature as $TF_r^{(t)}(i,j)$, which is computed with Eq.(2).

$$TF_r^{(t)}(i,j) = W_r \, \mathrm{ReLU}(H_{s,i}^{(t)} \circ H_{o,j}^{(t)}) + b_r \quad (2)$$

where $\circ$ denotes the *Hadamard Product* operation, ReLU is the activation function, $H_{s,i}^{(t)}$ and $H_{o,j}^{(t)}$ are the feature representations of tokens w_i and w_j at the t-th iteration respectively.

GFM Module This module mines the expected two kinds of global features, based on which new *subjects* and *objects* features are generated. Then these two new generated features will be fed back to *TFG* for next iteration. Specifically, this module consists of following three steps.

Step 1, to combine table features. Supposing current iteration is *t*, we first concatenate the table features of all relations together to generate an unified table feature (denoted as $TF^{(t)}$). And this unified table feature will contain the information of both token pairs and relations. Then we use a *max pooling* operation and an *FFN* model on $TF^{(t)}$ to generate a subject-related table feature ($TF_s^{(t)}$) and an object-related table feature ($TF_o^{(t)}$) respectively, as shown in Eq.(3).

$$\begin{aligned} TF_s^{(t)} &= W_s \underset{s}{\mathrm{maxpool}}\,(TF^{(t)}) + b_s \\ TF_o^{(t)} &= W_o \underset{o}{\mathrm{maxpool}}\,(TF^{(t)}) + b_o \end{aligned} \quad (3)$$

where $W_{s/o} \in \mathbb{R}^{(|L|\times|R|)\times d_h}$ are trainable weights, and $b_{s/o} \in \mathbb{R}^{d_h}$ are trainable biases.

Here the *max pooling* is used to highlight the important features that are helpful for the subject and object extractions respectively from a global perspective.

Step 2, to mine the expected two kinds of global features. Here we mainly use a *Transformer*-based model (Vaswani et al., 2017) to mine the global associations of relations and of token pairs.

First, we use a *Multi-Head Self-Attention* method on $TF_{s/o}^{(t)}$ to mine the global associations of relations. The self-attention mechanism can reveal the importance of an item from the perspective of other items, thus it is very suitable to mine the expected relation associations.

Then we mine the global associations of token pairs with a *Multi-Head Attention* method. The sentence representation *H* is also taken as part of input here. We think *H* may contain some global semantic information of a token to some extent since the input sentence is encoded as a whole, thus it is helpful for mining the global associations of token pairs from a whole sentence perspective.

Next, we generate new *subjects* and *objects* features with an *FFN* model.

In summary, the whole global association mining process can be written with following Eq.(4).

$$\begin{aligned} \hat{TF}_{s/o}^{(t)} &= \mathrm{MultiHeadSelfAtt}(TF_{s/o}^{(t)}) \\ \hat{H}_{(s/o)}^{(t+1)} &= \mathrm{MultiHeadAtt}(\hat{TF}_{s/o}^{(t)}, H, H) \\ H_{(s/o)}^{(t+1)} &= \mathrm{ReLU}(\hat{H}_{(s/o)}^{(t+1)} W + b) \end{aligned} \quad (4)$$

Step 3, to further tune the *subjects* and *objects* features generated in previous step.

One can notice that if we flat the iterative modules of *TFG* and *GFM*, our model would equal to a very deep network, thus it is possible to suffer from the *vanishing gradient* issue. To avoid this, we use a *residual network* to generate the final *subjects* and *objects* features, as written in Eq. (5).

$$H_{(s/o)}^{(t+1)} = \mathrm{LayerNorm}\,(H_{(s/o)}^{(t)} + H_{(s/o)}^{(t+1)}) \quad (5)$$

Finally, these *subjects* and *objects* features are fed back to the *TFG* module for next iteration. Note that the parameters of *TFG* and *GFM* are *shared* cross different iterations.

TG Module Taking the table features at the last iteration ($TF^{(N)}$) as input, this module outputs all the triples. Specifically, for each relation, its table is firstly filled with the method shown in Eq. (6).

$$\begin{aligned} \hat{table}_r(i,j) &= \mathrm{softmax}\,(TF_r^{(N)}(i,j)) \\ table_r(i,j) &= \underset{l \in L}{\mathrm{argmax}}\,(\hat{table}_r(i,j)[l]) \end{aligned} \quad (6)$$

where $\hat{table}_r(i,j) \in \mathbb{R}^{|L|}$, and $table_r(i,j)$ is the labeled result for the token pair (w_i, w_j) in the table of relation r.

Then, *TG* decodes the filled tables and deduces all triples with Algorithm 1. The main idea of our algorithm is to generate an entity pair set for each relation according to its filled table. And each entity pair in this set would correspond to a minimal

Algorithm 1 Table Decoding Strategy

Input: The relation set R, the sentence $S = \{w_1, w_2, ..., w_n\}$, and all $table_r \in \mathbb{R}^{n \times n}$ for each relation $r \in R$.
Output: The predicted triplet set, RT.

Define two temporary triple sets H and T, and initialize $H, T, RT \leftarrow \emptyset, \emptyset, \emptyset$.
for each $r \in R$ **do**
  Define three temporary sets WP_r^H, WP_r^T, and WP_r^S, which consist of token pairs whose ending tags in $table_r$ are "H", "T" and "S" respectively.
  for each $(w_i, w_j) \in WP_r^H$ **do** // forward search
    1) Find a token pair (w_k, w_m) from WP_r^T that satisfies: $i \leq k$, $j \leq m$, $table_r[(w_i, w_j)]$ and $table_r[(w_k, w_m)]$ match, (w_i, w_j) and (w_k, w_m) are closest in the table, and the number of words contained in subject $w_{i...k}$ and object $w_{j...m}$ are consistent with the corresponding tags.
    2) Add $(w_{i...k}, r, w_{j...m})$ to H.
  end for
  for each $(w_k, w_m) \in WP_r^T$ **do** // reverse search
    1) Find a token pair (w_i, w_j) from WP_r^H with a similar process as forward search.
    2) Add $(w_{i...k}, r, w_{j...m})$ to T.
  end for
  for each $(w_i, w_j) \in WP_r^S$ **do**
    Add (w_i, r, w_j) to RT
  end for
end for
$RT \leftarrow RT \cup H \cup T$
return RT

continuous token span in the filled table. Then each entity pair would form a triple with the relation that corresponds to the considered table. Specifically, in our decoding algorithm, we design three paralleled search routes to extract entity pairs of each relation. The first one (*forward search*, red arrows in Figure 1) generates entity pairs in an order of from head tokens to tail tokens. The second one (*reverse search*, green arrows in Figure 1) generates entity pairs in an order of from tail tokens to head tokens, which is designed mainly to handle the nested entities. And the third one (blue arrows in Figure 1) generates entity pairs that are single-token pairs.

Here we take the sentence shown in Figure 1 as a concrete sample to further explain our decoding algorithm. For example, in the demonstrated table, the token pair (*Edward*, *New*) has an "MMH" label, so the algorithm has to search forward to concatenate adjacent token pairs until a token pair that has a label "MMT" is found, so that to form the complete (*subject*, *object*) pair. And the *forward search* would be stopped when it meets the token pair (*Thomas*, *York*) that has the label "MMT". However, the formed entity pair (*Edward Thomas*, *New York*) is a wrong entity pair in the demonstrated example since the expected pair is (*Edward Thomas*, *New York City*). Such kind of errors are caused by the nested entities in the input sentence, like the "*New York*" and "*New York City*". These nested entities will make the *forward search* stops too early. In such case, the designed *reverse search* will play an important supplementary role. In the discussed example, the *reverse search* will first find the token pair (*Thomas*, *City*) that has an "MMT" label and has to further find a token pair that has an "MMH" label. Thus it will precisely find the expected entity pair (*Edward Thomas*, *New York City*).

Of course, if there are few nested entities in a dataset, the *reverse search* can be removed, which would be better for the running time. Here we leave it to make our model have a better generalization ability so that can be used in diverse datasets.

Category	NYT29		NYT24		WebNLG	
	Train	Test	Train	Test	Train	Test
Normal	53444	2963	37013	3266	1596	246
EPO	8379	898	9782	978	227	26
SEO	9862	1043	14735	1297	3406	457
ALL	63306	4006	56195	5000	5019	703
Relation	29		24		216 / 171*	

Table 1: Statistics of datasets. *EPO* and *SEO* refer to *entity pair overlapping* and *single entity overlapping* respectively (Zeng et al., 2018). Note that a sentence can belong to both *EPO* and *SEO*. And 216 / 171* means that there are 216 / 171 relations in WebNLG and WebNLG* respectively.

3.3 Loss Function

We define the model loss as follows.

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{r=1}^{|R|} -\log p\left(y_{r,(i,j)} = table_r(i,j)\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{r=1}^{|R|} -\log \hat{table}_r(i,j)[y_{r,(i,j)}]\end{aligned} \tag{7}$$

where $y_{r,(i,j)} \in [1, |L|]$ is the index of the ground truth label of (w_i, w_j) for the relaion r.

4 Experiments

4.1 Experimental Settings

Datasets We evaluate our model on three benchmark datasets: NYT29 (Takanobu et al., 2019), NYT24 (Zeng et al., 2018) and WebNLG (Gardent et al., 2017). Both NYT24 and WebNLG have two different versions according to following two annotation standards: 1) annotating the last token of

each entity, and 2) annotating the whole entity span. Different work chooses different versions of these datasets. To evaluate our model comprehensively, we use both kinds of datasets. For convenience, we denote the datasets based on the first annotation standard as NYT24* and WebNLG*, and the datasets based on the second annotation standard as NYT24 and WebNLG. Some statistics of these datasets are shown in Table 1.

Evaluation Metrics The standard micro precision, recall, and *F1* score are used to evaluate the results.

Note that there are two match standards for the RTE task: one is *Partial Match* that an extracted triplet is regarded as correct if the predicted relation and the head tokens of both subject entity and object entity are correct; and the other is *Exact Match* that a triple would be considered correct only when its entities and relation are completely matched with a correct triple. To fairly compare with existing models, we follow previous work (Wang et al., 2020; Wei et al., 2020; Sun et al., 2021) and use *Partial Match* on NYT24* and WebNLG*, and use *Exact Match* on NYT24, NYT29, and WebNLG.

In fact, since only one token of each entity in NYT24* and WebNLG* is annotated, the results of *Partial Match* and *Exact Match* on these two datasets are actually the same.

Baselines We compare our model with following strong state-of-the-art RTE models: *CopyRE* (Zeng et al., 2018), *GraphRel* (Fu et al., 2019), *CopyMTL* (Zeng et al., 2020), *OrderCopyRE* (Zeng et al., 2019), *ETL-Span* (Yu et al., 2019), *WDec* (Nayak and Ng, 2020), *RSAN* (Yuan et al., 2020), *RIN* (Sun et al., 2020), *CasRel* (Wei et al., 2020), *TPLinker* (Wang et al., 2020), *SPN* (Sui et al., 2020), and *PMEI* (Sun et al., 2021).

Most of the experimental results of these baselines are copied from their original papers directly. Some baselines did not report their results on some of the used datasets. In such case, we report the best results we obtained with the provided source code (if the source codes is available). For simplicity, we denote our model as *GRTE*, the abbreviation of *G*lobal feature oriented *RTE* model.

Implementation Details Adam (Kingma and Ba, 2015) is used to optimize *GRTE*. The learning rate, epoch and batch size are set to 3×10^{-5}, 50, 6 respectively. The iteration numbers (the hyperparameter N) on NYT29, NYT24*, NYT24, WebNLG* and WebNLG are set to 3, 2, 3, 2, and 4 respectively. Following previous work (Wei et al., 2020; Sun et al., 2021; Wang et al., 2020), we also implement a *BiLSTM*-encoder version of *GRTE* where 300-dimensional GloVe embeddings (Pennington et al., 2014) and 2-layer stacked BiLSTM are used. In this version, the hidden dimension of these 2 layers are set as 300 and 600 respectively. All the hyperparameters reported in this work are determined based on the results on the development sets. Other parameters are randomly initialized. Following *CasRel* and *TPLinker*, the max length of input sentences is set to 100.

4.2 Main Experimental Results

The main results are in the top two parts of Table 2, which show *GRTE* is very effective. On all datasets, it achieves almost all the best results in term of *F1* compared with the models that use the same kind of encoder (either the BiLSTM based encoder or the BERT based encoder). The only exception is on NYT24*, where the *F1* of $GRTE_{LSTM}$ is about 1% lower than that of $PMEI_{LSTM}$. However, on the same dataset, the *F1* score of $GRTE_{BERT}$ is about 2.9% higher than that of $PMEI_{BERT}$.

The results also show that *GRTE* achieves much better results on NYT29, NYT24 and WebNLG: its *F1* scores improve about 1.9%, 1.1%, and 3.3% over the previous best models on these three datasets respectively. Contrastively, its *F1* scores improve about 0.5% and 0.5% over the previous best models on NYT24* and WebNLG* respectively. This is mainly because that *GRTE* could not realize its full potential on NYT24* and WebNLG* where only one token of each entity is annotated. For example, under this annotation standard, except "N/A", "SSH", and "SST", all the other defined labels in *GRTE* are redundant. But it should be noted that the annotation standard on NYT24* and WebNLG* simplifies the RTE task, there would not be such a standard when a model is really deployed. Thus, the annotation standard on NYT29, NYT24 and WebNLG can better reveal the true performance of a model. Accordingly, *GRTE*'s better performance on them is more meaningful.

We can further see that compared with the previous best models, *GRTE* achieves more performance improvement on WebNLG than on other datasets. For example, $GRTE_{LSTM}$ even outperforms all other compared baselines on WebNLG, including those models that use *BERT*. We think this is mainly because that the numbers of relations in WebNLG are far more than those of in NYT29 and

Model	NYT29			NYT24*			NYT24			WebNLG*			WebNLG		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
CopyRE	–	–	–	61.0	56.6	58.7	–	–	–	37.7	36.4	37.1	–	–	–
GraphRel	–	–	–	63.9	60.0	61.9	–	–	–	44.7	41.1	42.9	–	–	–
OrderCopyRE	–	–	–	77.9	67.2	72.1	–	–	–	63.3	59.9	61.6	–	–	–
ETL-Span	74.5⋆	57.9⋆	65.2⋆	84.9	72.3	78.1	85.5	71.7	78.0	84.0	91.5	87.6	84.3	82.0	83.1
WDec	77.7	60.8	68.2	–	–	–	88.1	76.1	81.7	–	–	–	–	–	–
RSAN	–	–	–	–	–	–	85.7	83.6	84.6	–	–	–	80.5	83.8	82.1
RIN	–	–	–	87.2	87.3	87.3	83.9	85.5	84.7	87.6	87.0	87.3	77.3	76.8	77.0
CasRel$_{LSTM}$	–	–	–	84.2	83.0	83.6	–	–	–	86.9	80.6	83.7	–	–	–
PMEI$_{LSTM}$	–	–	–	88.7	86.8	87.8	84.5	84.0	84.2	88.7	87.6	88.1	78.8	77.7	78.2
TPLinker$_{LSTM}$	–	–	–	83.8	83.4	83.6	86.0	82.0	84.0	90.8	90.3	90.5	91.9	81.6	86.4
CasRel$_{BERT}$	77.0⋆	68.0⋆	72.2⋆	89.7	89.5	89.6	89.8⋆	88.2⋆	89.0⋆	93.4	90.1	91.8	88.3⋆	84.6⋆	86.4⋆
PMEI$_{BERT}$	–	–	–	90.5	89.8	90.1	88.4	88.9	88.7	91.0	92.9	92.0	80.8	82.8	81.8
TPLinker$_{BERT}$	78.0*	68.1*	72.7*	91.3	92.5	91.9	91.4	92.6	92.0	91.8	92.0	91.9	88.9	84.5	86.7
SPN$_{BERT}$	76.0*	71.0*	73.4*	**93.3**	91.7	92.5	92.5	92.2	92.3	93.1	93.6	93.4	85.7⋆	82.9⋆	84.3⋆
GRTE$_{LSTM}$	74.3	67.9	71.0	87.5	86.1	86.8	86.2	87.1	86.6	90.1	91.6	90.8	88.0	86.3	87.1
GRTE$_{BERT}$	**80.1**	**71.0**	**75.3**	92.9	**93.1**	**93.0**	**93.4**	**93.5**	**93.4**	**93.7**	**94.2**	**93.9**	**92.3**	**87.9**	**90.0**
GRTE$_{w/o\ GFM}$	77.9	68.9	73.1	90.6	92.5	91.5	91.8	92.6	92.2	92.4	91.1	91.7	88.4	86.7	87.5
GRTE$_{GRU\ GFM}$	78.2	**71.7**	74.8	92.5	92.9	92.7	93.4	92.2	92.8	93.4	92.6	93.0	90.1	**88.0**	89.0
GRTE$_{w/o\ m-h}$	77.8	70.9	74.2	91.9	92.9	92.4	93.2	92.9	93.0	92.9	92.1	92.5	90.5	87.6	89.0
GRTE$_{w/o\ shared}$	79.5	71.5	**75.3**	92.7	93.0	92.8	**93.6**	92.7	93.1	93.4	94.0	93.7	91.5	87.4	89.4

Table 2: Main results. A model with a subscript *LSTM* refers to replace its *BERT* based encoder with the *BiLSTM* based encoder. ⋆ means the results are produced by us with the available source code.

NYT24 (see Table 1), which means there are more global associations of relations can be mined. Generally, the more relations and entities there are in a dataset, the more global correlations there would be among triples. Accordingly, our model could perform more better on such kind of datasets than other local features based methods. For example, the number of relations in WebNLG is almost 7 times of those in NYT, and *GRTE* achieves much more performance improvement over the compared baselines on WebNLG than on NYT.

4.3 Detailed Results

In this section, we conduct detailed experiments to demonstrate the effectiveness of our model from following two aspects.

First, we conduct some ablation experiments to evaluate the contributions of some main components in *GRTE*. To this end, we implement following model variants.

(i) GRTE$_{w/o\ GFM}$, a variant that removes the *GFM* module completely from *GRTE*, which is to evaluate the contribution of *GFM*. Like previous table filling based methods, GRTE$_{w/o\ GFM}$ extracts triples only based on local features.

(ii) GRTE$_{GRU\ GIF}$, a variant that uses *GRU* (taking H and $TF^{(t)}_{s/o}$ as input) instead of *Transformer* to generate the results in Eq. (4), which is to evaluate the contribution of *Transformer*.

(iii) GRTE$_{w/o\ m-h}$, a variant that replaces the *multi-head attention* method in *GFM* with a *single-head attention* method, which is to evaluate the contribution of the *multi-head attention*.

(iv) GRTE$_{w/o\ shared}$, a variant that uses different parameters for the modules of *TFG* and *GFM* at different iterations, which is to evaluate the contribution of the parameter share mechanism.

All these variants use the *BERT*-based encoder. And their results are shown in the bottom part of Table 2, from which we can make following observations.

(1) The performance of GRTE$_{w/o\ GFM}$ drops greatly compared with *GRTE*, which confirms the importance of using two kinds of global features for table filling. We can further notice that on NYT29, NYT24, and WebNLG, the *F1* scores of GRTE$_{w/o\ GFM}$ increases by 0.4%, 0.4%, and 0.8% respectively over *TPLinker*. Both *TPLinker* and GRTE$_{w/o\ GFM}$ extract triples based on local features, and the main difference between them is the table filling strategy. So these results prove the effectiveness of our table filling strategy. The *F1* scores of GRTE$_{w/o\ GFM}$ on NYT24* and WebNLG* are slightly lower than those of *TPLinker*, as explained above, this is because each entity in NYT24* and WebNLG*, only one token is annotated for each entity, GRTE$_{w/o\ GFM}$ could not realize its full potential.

Model	NYT24*								WebNLG*							
	Normal	SEO	EPO	T = 1	T = 2	T = 3	T = 4	T $\geq$ 5	Normal	SEO	EPO	T = 1	T = 2	T = 3	T = 4	T $\geq$ 5
CasRel$_{BERT}$	87.3	91.4	92	88.2	90.3	91.9	94.2	83.7	89.4	92.2	94.7	89.3	90.8	94.2	92.4	90.9
TPLinker$_{BERT}$	90.1	93.4	94.0	90.0	92.8	93.1	96.1	90.0	87.9	92.5	95.3	88.0	90.1	94.6	93.3	91.6
SPN$_{BERT}$	90.8	94.0	94.1	**90.9**	93.4	94.2	95.5	90.6	89.5*	94.1*	90.8*	89.5	91.3	96.4	94.7	93.8
GRTE$_{BERT}$	**91.1**	**94.4**	**95**	90.8	**93.7**	**94.4**	**96.2**	**93.4**	**90.6**	**94.5**	**96**	**90.6**	**92.5**	**96.5**	**95.5**	**94.4**

Table 3: F1 scores on sentences with different overlapping pattern and different triplet number. Results of *CasRel* are copied from *TPLinker* directly. "T" is the number of triples contained in a sentence. * means the results are produced by us with the provided source codes.

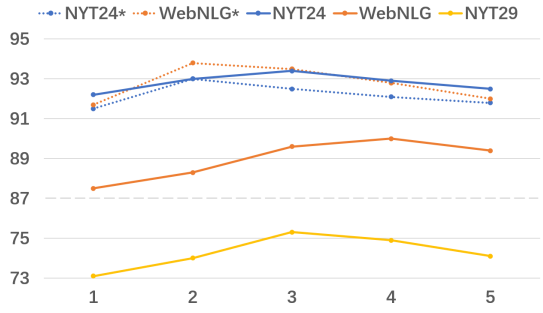

Figure 3: F1 results under different *N*.

(2) The performance of GRTE$_{GRU\ GFM}$ drops compared with *GRTE*, which indicates *Transformer* is more suitable for the global feature mining than *GRU*. But even so, we can see that on all datasets, GRTE$_{GRU\ GFM}$ outperforms almost all previous best models and GRTE$_{w/o\ GFM}$ in term of *F1*, which further indicates the effectiveness of using global features.

(3) The results of GRTE$_{w/o\ m-h}$ are lower than those of *GRTE*, which shows the *multi-head attention* mechanism plays an important role for global feature mining. In fact, the importance of different features is different, the *multi-head attention* mechanism performs the feature mining process from multiple aspects, which is much helpful to highlight the more important ones.

(4) The results of GRTE$_{w/o\ shared}$ are slightly lower than those of *GRTE*, which shows the share mechanism is effective. In fact, the mechanism of using distinct parameters usually works well only when the training samples are sufficient. But this condition is not well satisfied in RTE since the training samples of a dataset are not sufficient enough to train too many parameters.

Second, we evaluate the influence of the iteration number N. The results are shown in Figure 3, from which following observations can be made.

(1) On NYT24* and WebNLG*, the annotation standard is relatively simple. So *GRTE* achieves the best results with two iterations. But on NYT29, NYT24, and WebNLG, more iterations are usually required. For example, *GRTE* achieves the best results when *N* is 3, 3, and 4 respectively on them.

(2) On all datasets, *GRTE* gets obvious performance improvement (even the maximum performance improvement on some datasets) at $N = 2$ where *GFM* begins to play its role , which indicates again that using global features can significantly improve the model performance.

(3) *GRTE* usually achieves the best results within a small number of iterations on all datasets including WebNLG or WebNLG* where there are lots of relations. In fact, *GRTE* outperforms all the pervious best models even when *N* = 2. This is a very important merit because it indicates that even used on some datasets where the numbers of relations are very large, the *efficiency* would not be a burden for *GRTE*, which is much meaningful when *GRTE* is deployed in some real scenarios.

4.4 Analyses on Different Sentence Types

Here we evaluate *GRTE*'s ability for extracting triples from sentences that contain overlapping triples and multiple triples. For fair comparison with the previous best models (*CasRel*, *TPLinker*, and *SPN*), we follow their settings which are: (i) classifying sentences according to the degree of overlapping and the number of triples contained in a sentence, and (ii) conducting experiments on different subsets of NYT24* and WebNLG*.

The results are shown in Table 3. We can see that: (i) *GRTE* achieves the best results on all three kinds of overlapping sentences on both datasets, and (ii) *GRTE* achieves the best results on almost all kinds of sentences that contain multiple triples. The only exception is on NYT24* where the *F1* score of *GRTE* is slightly lower than that of *SPN* when *T* is 1. The main reason is that there are less associations among token pairs when *T* is 1, which slightly degrades the performance of *GRTE*.

Model	NYT24*			WebNLG*		
	Params_{all}	$\text{Prop}_{encoder}$	Inference Time	Params_{all}	$\text{Prop}_{encoder}$	Inference Time
CasRel_{BERT}	107,719,680	99.96%	53.9	107,984,216	99.76%	77.5
TPLinker_{BERT}	109,602,962	98.82%	18.1 / 83.5†	110,281,220	98.21%	26.9 / 120.4†
SPN_{BERT}	141,428,765	76.58%	26.4 / 107.9†	150,989,744	71.73%	22.6 / 105.7†
GRTE_{BERT}	119,387,328	90.72%	21.3 / 109.6†	122,098,008	88.70%	28.7 / 124.1†

Table 4: Computational efficiency. Params_{all} is the number of parameters for the entire model. $\text{Prop}_{encoder}$ refers to the proportion of encoder parameters in the total model parameters. Inference Time represents the average time (millisecond) the model takes to process a sample. † marks the inference time when the batch size is set to 1.

In fact, *GRTE* maintains a table for each relation, and the *TG* module extracts triples for each relation independently. Thus it can well handle above two kinds of complex sentences by nature.

4.5 Analyses on Computational Efficiency

Table 4 shows the comparison results of computational efficiency between *GRTE* and some previous best models. To be fair, we follow the settings in *TPLinker*: analyze the parameter scale and the inference time on NYT* and WebNLG*. All the results are obtained by running the compared models on a TitanXP, and the batch size is set to 6 for all models that can be run in a batch mode.

The parameter number of *GRTE* is slightly larger than that of *TPLinker*, which is mainly due to the using of a *Transformer*-based model. But when compared with *SPN* that uses the *Transformer* model too, we can see that *GRTE* has a smaller number of parameters due to its parameter share mechanism.

We can also see that *GRTE* achieves a very competitive inference speed. This is mainly because of following three reasons. First, *GRTE* is a one-stage extraction model and can process samples in a batch mode (*CasRel* can only process samples one by one). Second, as analyzed previously, it has an efficient table filling strategy that needs to fill fewer table items. Third, as analyzed previously, *GRTE* often achieves the best results within a small number of iterations, thus the iteration operations will not have too much impact on the inference speed of *GRTE*.

In fact, as *TPLinker* pointed out that for all the models that use BERT (or other kinds of pre-trained language models) as their basic encoders, BERT is usually the most time-consuming part and takes up the most of model parameters, so the time cost of other components in a model is not significant.

Besides, there is another important merit of our model: it needs less training time than existing state-of-the-art models like *CasRel*, *TPLinker*, and *SPN* etc. As pointed out previously, the epoch of our model on all datasets is 50. But on the same datasets, the epochs of all the mentioned models are 100. From Table 4 we can see that all these models have similar inference speed. For each model, the training speed of each epoch is very close to its inference speed (during training, there would be extra time cost for operations like the back propagation), thus we can easily know that our model needs less time for training since our model has a far less epoch number.

5 Conclusions

In this study, we propose a novel table filling based RTE model that extracts triples based on two kinds of global features. The main contributions of our work are listed as follows. First, we make use of the global associations of relations and of token pairs. Experiments show these two kinds of global features are much helpful for performance. Second, our model works well on extracting triples from complex sentences containing overlapping triples or multiple triples. Third, our model is evaluated on three benchmark datasets. Extensive experiments show that it consistently outperforms all the compared strong baselines and achieves state-of-the-art results. Besides, our model has a competitive inference speed and a moderate parameter size.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61572120 and No.U1708261), the Fundamental Research Funds for the Central Universities (No.N181602013 and No.N2016006), Shenyang Medical Imaging Processing Engineering Technology Research Center (17-134-8-00), Ten Thousand Talent Program (No.ZX20200035), and Liaoning Distinguished Professor (No.XLYC1902057).

References

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Markus Eberts and Adrian Ulges. 2019. Span-based joint entity and relation extraction with transformer pre-training. In *ECAI*, pages 2006–2013.

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

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188.

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

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

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

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

Tapas Nayak and Hwee Tou Ng. 2020. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8528–8535. AAAI Press.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, Xiangrong Zeng, and Shengping Liu. 2020. Joint entity and relation extraction with set prediction networks. *CoRR*, abs/2011.01675.

Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao, and Xudong Liu. 2020. Recurrent interaction network for jointly extracting entities and classifying relations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3722–3732. Association for Computational Linguistics.

Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao, and Xudong Liu. 2021. Progressive multitask learning with controlled information flow for joint entity and relation extraction. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):7072–7079.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

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

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging

framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1476–1488, Online. Association for Computational Linguistics.

Bowen Yu, Zhenyu Zhang, Xiaobo Shu, Tingwen Liu, Yubin Wang, Bin Wang, and Sujian Li. 2019. Joint extraction of entities and relations based on a novel decomposition strategy. In *ECAI*, pages 2282–2289.

Yue Yuan, Xiaofei Zhou, Shirui Pan, Qiannan Zhu, Zeliang Song, and Li Guo. 2020. A relation-specific attention network for joint entity and relation extraction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, volume 4, pages 4054–4060.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3(6):1083–1106.

Daojian Zeng, Haoran Zhang, and Qianying Liu. 2020. Copymtl: Copy mechanism for joint extraction of entities and relations with multi-task learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(5):9507–9514.

Xiangrong Zeng, Shizhu He, Daojian Zeng, Kang Liu, Shengping Liu, and Jun Zhao. 2019. Learning the extraction order of multiple relational facts in a sentence with reinforcement learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 367–377.

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

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1730–1740, Copenhagen, Denmark. Association for Computational Linguistics.

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

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434.