Jointly Extracting Multiple Triplets with Multilayer Translation Constraints

Zhen Tan,[1] **Xiang Zhao,**[1,3,*] **Wei Wang,**[2,4] **Weidong Xiao**[1,3]
[1]Key Laboratory of Science and Technology on Information System Engineering,
National University of Defense Technology, China
[2]School of Computer Science and Engineering, UNSW, Australia
[3]Collaborative Innovation Center of Geospatial Technology, China
[4]College of Computer Science and Technology, DGUT, China

Abstract

Triplets extraction is an essential and pivotal step in automatic knowledge base construction, which captures structural information from unstructured text corpus. Conventional extraction models use a pipeline of named entity recognition and relation classification to extract entities and relations, respectively, which ignore the connection between the two tasks. Recently, several neural network-based models were proposed to tackle the problem, and achieved state-of-the-art performance. However, most of them are unable to extract multiple triplets from a single sentence, which are yet commonly seen in real-life scenarios. To close the gap, we propose in this paper a joint neural extraction model for multi-triplets, namely, TME, which is capable of adaptively discovering multiple triplets simultaneously in a sentence via ranking with translation mechanism. In experiment, TME exhibits superior performance and achieves an improvement of 37.6% on F1 score over state-of-the-art competitors.

1 Introduction

Triplets extraction captures structural information, i.e., triples of two entities with one relation, from unstructured text corpus, which is an essential and pivotal step in automatic knowledge base construction (Bollacker et al. 2008). Conventional models use a pipeline of named entity recognition (NER) (Shaalan 2014) and relation classification (RC) (Rink and Harabagiu 2010) to extract entities and relations, respectively, to produce the final triplets. Such pipelined methods may not fully capture and exploit correlations between the NER and RC tasks, being susceptible to cascading errors (Li and Ji 2014).

To overcome the shortcoming, recent research resorted to joint models, most of which are features-based structured models (Kate and Mooney 2010; Yu and Lam 2010; Chan and Roth 2011; Miwa and Sasaki 2014), which require excessive manual intervention and supervised natural language processing toolkits to construct *multiplex* and *complicated* features. Lately, several neural models have been presented to jointly extract entities and relations. Specifically, Zheng et al. utilized Bi-LSTM to learn shared hidden features, then used LSTM to extract entities, and CNN for relations (Zheng et al. 2017a). Miwa and Bansal used an end-to-end model to extract entities, and dependency tree was harnessed to determine relations (Miwa and Bansal 2016). These two models first recognize entities, and then choose a semantic relation for *every possible* pair of extracted entities; in this case, the RC classifier has a comparatively low precision but high recall, since it is misled by many of the pairs that fall into the `other` category [1].

Meanwhile, there are models that extract confined appearances of target relations. In particular, Zheng et al. transformed joint extraction into a tagging problem to tag entities and relations in a unified tagging scheme, and utilized an end-to-end model to solve the problem (Zheng et al. 2017b). Nevertheless, in this model each entity is constrained to be involved in *only one* relation in every sentence. Katiyar and Cardie also used Bi-LSTM to extract entities, and a self-attention mechanism was incorporated to extract relations (Katiyar and Cardie 2017). The model assumes that an entity could relate to *only one* of its *preceding* entities in the sentence.These two models still have not fully recognized and attached importance to the fact that there could be multiple relations associated with an entity; in this case, the RC task performs at comparatively high precision but low recall, since the scope of candidates for RC is confined.

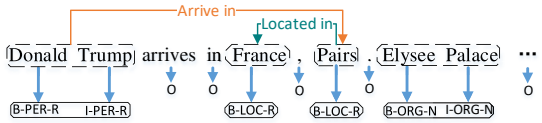

Figure 1: Sample Sentence with Tri-part Tagging

To sum up, existing joint models either extract limited relations with unpragmatic constraints (one relation for one sentence, or relating to only one preceding entity), or simply produce too many candidates for RC (relations for all possible entity pairs). Thorough investigation suggests that the main reason lies in that they overlooked the impact of

*Corresponding Author: Xiang Zhao (xiangzhao@nudt.edu.cn)

[1]In RC, class `other` means that there is no semantic relation between entities, or it is out of the given set of target relations (Rink and Harabagiu 2010).

multi-triplets, which are commonly seen in real-life large corpus [2]. Let us consider the news flash sentence in Figure 1. It can be seen that there are two relations associated with the entity `Paris`, i.e., (`Donald Trump`, `Arrive in`, `Paris`) and (`Paris`, `Located in`, `France`) in triplet form. Nevertheless, all the aforementioned models fail to capture them entirely. In particular, the model of (Zheng et al. 2017b) assumes that the entity `Paris` belongs to only one triplet, and hence, either of the two triplets would be concealed. The model of (Katiyar and Cardie 2017) finds relations between an entity and one entity preceding it, in which case either of the relation from `Paris` to `Donald Trump` or `France` would not be discovered. On the other hand, the models of (Miwa and Bansal 2016; Zheng et al. 2017a) presume that every entity pair has a relation. Under this scenario, abundant pairs should be thrown into `other` class, but the features of `other` are rather difficult to learn during RC training; hence, the noisy entities (`Elysee Palace`) and unintended relations between (`Donald Trump`, `Elysee Palace`) further confuse the classifier. Thus, target relations may not be correctly detected or chosen for multi-triplets.

This paper aims to close the gap by recognizing and solving the problem of *multi-triplets extraction*. Intuitively, a good multi-triplets extractor can (1) judiciously distinguish the candidate entities that may be involved with target relations; (2) learn the complete features of entities and relations of every sentence; and (3) alleviate the impact of `other` relations on RC and also enhance the training of the extractor. To this end, we propose a novel joint extraction model for multi-triplets, namely, TME, which implements these ideas. To distinguish candidate entities and exclude irrelevant ones, we first design a tri-part tagging scheme using *position*, *type* and *relation* parts to describe the features of each word in a sentence (exemplified in Figure 1). To perform the tagging, we utilize Bi-LSTM+CRF to learn entity features by Bi-LSTM, and then to generate tag sequences by CRF for the words in the sentence, such that only entities that are likely to participate in target relations are identified. Afterwards for relation extraction, we use external sentence-irrelevant embeddings to describe relation features via embedding translation; that is, we require entities and relations to form triplet (e_h, e_t, r), satisfying translation-based constraint $e_h + r \approx e_t$. To prevent deviation of entity features from Bi-LSTM, we also enforce them to satisfy two additional constraints $\overrightarrow{e_h} + r \approx \overrightarrow{e_t}$ and $\overleftarrow{e_h} + r \approx \overleftarrow{e_t}$ (sketched in Figure 2). To further alleviate the impact of "`other` relations", we leverage a ranking-based extractor, where we only rank candidate relations in the relation list and the correct triplet is expected to be ranked high. To better train the joint model, we introduce a negative sampling strategy to enable a robust ranking-based relation extractor.

Contribution. In summary, we propose to investigate a novel problem of multi-triplets extraction, which is of practical significance but was largely overlooked, and the contribution of the paper is at least four-fold:

[2]For instance, in New York Times dataset (Riedel, Yao, and McCallum 2010), 37.4% sentences embody multi-triplets.

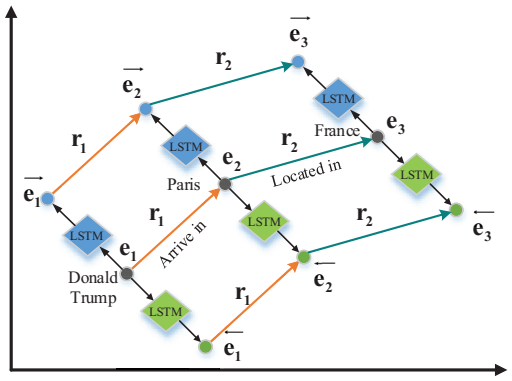

Figure 2: Multi-layer Embedding Translation

- We present a joint multi-triplets extractor TME, which employs a novel multi-layer model of embedding translation that tries to preserve relation features that an entity possesses during triplets extraction;
- We devise a tailored tri-part tagging scheme that scrupulously distinguishes candidate entities, which helps reduce noise from irrelevant entities;
- We propose to perform relation extraction by ranking candidate relations, while enforcing translation-based constraints using designated relation feature vectors;
- Trained with negative sampling, TME is demonstrated through comprehensive experiments to outperform its competitors on both single and multi-triplets extraction.

Organization. We first discuss related work, then introduce the framework and preliminaries. Afterwards, the model details are presented, followed by experiments. In the end, we conclude the paper with major findings.

2 Related Work

We discuss related work from three aspects: joint triplets extraction models and representative methods for NER and RC.

Joint Triplets Extraction. Besides the joint models (Miwa and Bansal 2016; Zheng et al. 2017a; 2017b; Katiyar and Cardie 2017; Ren et al. 2017) that we have reviewed in the introduction, there are several other related efforts towards joint triplets extraction. Roth and Yih (Roth and Yih 2004) and Yang et al. (Yang and Cardie 2013) proposed integer linear programming models to tackle the problem. Kate and Mooney (Kate and Mooney 2010) used card-pyramid parsing to jointly extract entities and relations, while Singh et al. leveraged on a probabilistic graphical model (Singh et al. 2013). These *feature-based* models require a lot of manually designed features, and thus, are difficult to be applied in large-scale applications.

Another stream of research employs a pipelined method of two sub-tasks NER and RC to accomplish triplets extraction. Due to the design flaw of the method, it tends to propagate errors between tasks, and hence, affects the overall precision and accuracy. Nonetheless, following briefs the representative models for NER and RC, respectively.

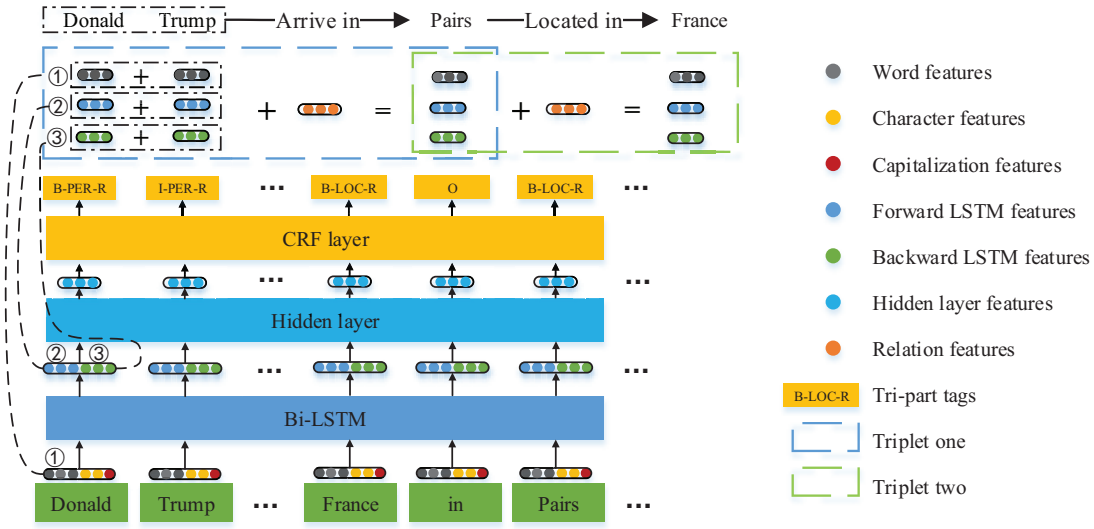

Figure 3: Framework of Joint Multi-triplets Extraction Model TME

Named Entity Recognition. Most NER models are either statistical or neural network-based. Hidden Markov models (Passos, Kumar, and McCallum 2014) and CRF (McCallum and Li 2003; Finkel, Grenager, and Manning 2005) are typical statistical models, which consider the correlation among the tags of entities. Neural network-based models (Hammerton 2003) pull out hidden features from each word for recognizing entities. Lately, neural networks combining CRF has been shown to achieve state-of-the-art results, such as CNN+CRF (Collobert et al. 2011) and BiLSTM+CRF (Lample et al. 2016).

Relation Classification. Classical feature-based (Rink and Harabagiu 2010) and kernel-based (Zhang, Zhang, and Su 2006) methods are often used to extract relations, which, however, suffer from low effectiveness and poor generalization. Hence, neural network-based methods were proposed, e.g., CNN (Santos, Xiang, and Zhou 2015) and RNN (including GRU and LSTM) (Zhang and Wang 2015). In order to capture both word sequence and dependency information, attention mechanism (Shen and Huang 2016; Wang et al. 2016) and shortest dependency tree (Xu et al. 2015) were leveraged to boost the performance.

3 Framework and Preliminaries

This section introduces the proposed framework for multi-triplets extraction, as well as some preliminaries.

3.1 Framework

A rationale that underlies many existing joint extraction models is that *if a sentence contains more than two entities, it is possible that there is one relation existing between any pair of entities.* Naturally, it suggests the following paradigm: given a sentence, we extract first entities (Stage I), and then relations between each candidate pairs of entities by classification (Stage II). However, the paradigm is flawed and performance deteriorates, when an entity has no relation, unwanted or multiple relations with others.

To resolve the issue, we present a revised framework (depicted in Figure 3) that generates candidate entity pairs without unpragmatic constraints in Stage I and refrains excessive irrelevant entities from going into Stage II. It comprises

- a neural model using Bi-LSTM+CRF to obtain entity features, which are reused through feature sharing by a multi-layer module for capturing complex relation features via translation mechanism;
- a tri-part tagging scheme for distinguishing whether an entity is involved with a wanted relation or not; and
- a margin-based relation ranker, trained with negative samples, for discovering appropriate relations between entity pairs.

3.2 Preliminaries

Bi-LSTM+CRF is a standardized architecture for recognizing entities, which has been shown to perform well on many NER tasks (Lample et al. 2016). It captures the dependencies among different words in a sentence, which consists of three layers—embedding layer, Bi-LSTM and CRF.

Embedding Layer. For an input sequence $\mathcal{X} = (x_1, x_2, \ldots, x_s)$, where s is the word length $|\mathcal{X}|$, we construct an input embedding $\mathbf{i}$ for every word $x \in \mathcal{X}$, which consists of three parts, i.e., word embedding $\mathbf{w}$, character-level embedding $\mathbf{c}_h$ and capitalization embedding $\mathbf{c}_a$. The dimension of $\mathbf{w}$, $\mathbf{c}_h$ and $\mathbf{c}_a$ are d_w, d_o and d_c, respectively. The purpose of introducing a character-level embedding is to comprehensively extract character features of words, which is accomplished by a character-driven Bi-LSTM. In particular, the character sequence of word x is fed into the Bi-LSTM, which produces a vector $\mathbf{c}_h$ such that $\mathbf{c}_h$ is the n-th output vector of the character-driven Bi-LSTM, where n is the character length of x.

Bi-LSTM Layer. A Bi-LSTM is used to capture sequence features. The input is an embedding sequence $\mathcal{I} = (\mathbf{i}_1, \ldots, \mathbf{i}_t, \ldots, \mathbf{i}_s)$ from the embedding layer, where $\mathbf{i}_t \in \mathbb{R}^{d_i}$ is d_i-dimensional such that $d_i = d_w + d_o + d_c$. In Bi-LSTM, there are two LSTMs: one extracts forward hidden features $\overrightarrow{\mathbf{h}_t}$ and the other extracts backward hidden features $\overleftarrow{\mathbf{h}_t}$. The function of Bi-LSTM is formulated as

$$\overrightarrow{\mathbf{h}}_t, \overrightarrow{\mathbf{c}}_t = \text{LSTM}(\mathbf{i}_t, \overrightarrow{\mathbf{h}}_{t-1}, \overrightarrow{\mathbf{c}}_{t-1}),$$
$$\overleftarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{c}}_t = \text{LSTM}(\mathbf{i}_t, \overleftarrow{\mathbf{h}}_{t+1}, \overleftarrow{\mathbf{c}}_{t+1}).$$

The output is $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}, \overleftarrow{\mathbf{h}_t}]$; we resort to an activation function to merge $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$, and project them into a d_l-dimensional space. That is,

$$\mathbf{l} = \texttt{tanh}(\mathbf{W}_l \, \texttt{tanh}(\mathbf{W}_t \mathbf{h}_t + \mathbf{b}_t) + \mathbf{b}_l),$$

where d_l is the number of distinct tags, $\mathbf{W}_l$ and $\mathbf{W}_t$ are matrices, $\mathbf{b}_t$ and $\mathbf{b}_l$ are biases.

CRF Layer. We use a CRF to decide tags for each output y_t. For a sentence $\mathcal{X}$, the input matrix $\mathbf{L}$ is the output of Bi-LSTM layer such that $\mathbf{L} = [\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_{s-1}, \mathbf{l}_s]^{\mathrm{T}}$, whose size is $s \times d_l$. Let $L_{i,j}$ denote the probability score of the j-th tag of the i-th word in the sentence. For a prediction sequence $\mathbf{y} = (y_1, y_2, \ldots, y_{s-1}, y_s)$, we define CRF score as

$$f(\mathbf{X}, \mathbf{y}) = \sum_{i=0}^{s+1} T_{y_i, y_{i+1}} + \sum_{i=1}^{s} L_{i, y_i},$$

where $\mathbf{T}$ is a transition matrix, $T_{i,j}$ denotes the transition score from the tag i to tag j, and y_0 and y_{s+1} are the starting and ending tags, respectively. Then, `softmax` is used to calculate the probability of the sequence $\mathbf{y}$ by

$$p(\mathbf{y}|\mathbf{X}) = \frac{e^{f(\mathbf{X}, \mathbf{y})}}{\sum_{\tilde{\mathbf{y}} \in \mathbf{Y}} e^{f(\mathbf{X}, \tilde{\mathbf{y}})}},$$

where $\mathbf{Y}$ denotes all possible tag sequences for $\mathbf{X}$.

In the framework, we leverage Bi-LSTM+CRF to perform tri-part tagging (to be introduced), which can recognize only entities that are likely to take part in some relation(s); that is, the output of the CRF layer is predicted tags containing information about whether a word is part of an entity, and simultaneously whether it relates to some relation(s).

4 Modules of Proposed Model

In this section, we detail the new modules of our proposed model, which comprises a tri-part tagging scheme for obtaining entity features, a multi-layer translation mechanism for capturing relation features, and a margin-based relation ranker trained with negative sampling.

4.1 Tri-part Tagging Scheme

Inspired by (Zheng et al. 2017b), we propose a tri-part tagging scheme (TTS) on the basis of Bi-LSTM+CRF, in order to give each word in a sentence a unique tag, which is used to extract entity features. It is constituted of three parts:

- In position part (**PP**), we use "BIO" to encode the position information of the words regarding an entity: "B" indicates that the word locates in the first place of an entity; "I" indicates it locates in a place after the first of an entity; and "O" indicates it locates in a non-entity place.
- In type part (**TP**), we associate words with type information of entities, e.g., in Figure 1, "PER", "LOC" and "ORG" denote a person, a location, and an organization, respectively.
- In relation part (**RP**), we annotate whether an entity in the sentence is involved in any relation: "R" indicates that the entity is involved in some relation(s) in the sentence; and "N" denotes that it does not participate in any wanted relation.

A sample result of TTS is provided in Figure 1, where the sentence contains four entities and two target relations. Specifically, `Donald` is in the first place of entity *Donald Trump* which has the type `Person`, and has relation with other entities. Thus, TTS tag of `Donald` is "B-PER-R". Similarly, the tag of `Trump` is "I-PER-R".

Remark. Compared with the classic BILOU tagging scheme (Li and Ji 2014; Miwa and Bansal 2016), TTS is conceived to describe position, type and relation information of each entity simultaneously. The major advantage of TTS is that while recognizing entities from sentences, it also removes noisy entities and facilitates multi-triplets extraction. This is deemed as the major difference from that of (Zheng et al. 2017b), rendering it more superior.

4.2 Multi-layer Translation Mechanism

Inspired by translation mechanism, we construct a multi-layer model for capturing relation features.

Notation. For an input sequence $\mathcal{X}$, $\mathcal{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_s)$ is the word embedding sequence, $\overrightarrow{\mathcal{H}} = (\overrightarrow{\mathbf{h}_1}, \overrightarrow{\mathbf{h}_2}, \ldots, \overrightarrow{\mathbf{h}_s})$ is the output of forward LSTM and $\overleftarrow{\mathcal{H}} = (\overleftarrow{\mathbf{h}_1}, \overleftarrow{\mathbf{h}_2}, \ldots, \overleftarrow{\mathbf{h}_s})$ is the output of reverse LSTMs. $\mathcal{T}$, $\mathcal{E}$ and $\mathcal{R}$ denote triplet set, entity set and relation set, respectively; t denotes a triplet $(e_1, e_2, r) \in \mathcal{T}$, where $e_1, e_2 \in \mathcal{E}$ and $r \in \mathcal{R}$. For an entity in $\mathcal{X}$, $e = (x_i, \ldots, x_{i+j}, \ldots, x_{i+e_l})$, where i denotes *starting* position in $\mathcal{X}$, j denotes the j-th word in the entity, e_l is the length of entity. We sum up the embeddings in the position of entity to represent the entity embeddings, and

$$\mathbf{e} = \sum_{k=i}^{i+e_l} \mathbf{w}_k, \overrightarrow{\mathbf{e}} = \sum_{k=i}^{i+e_l} \overrightarrow{\mathbf{h}}_k, \overleftarrow{\mathbf{e}} = \sum_{k=i}^{i+e_l} \overleftarrow{\mathbf{h}}_k,$$

where $\mathbf{e}$, $\overrightarrow{\mathbf{e}}$ and $\overleftarrow{\mathbf{e}}$ are entity embeddings in embedding layer and Bi-LSTM layer, respectively.

Model. For each triplet $t = (e_1, e_2, r) \in \mathcal{T}$ in the sequence, we obtain the head entity embedding $\mathbf{e_1}$ and tail entity embedding $\mathbf{e_2}$ in the embedding layer, and generate a corresponding relation embedding $\mathbf{r}$. We require that $\mathbf{e_1}$ adding $\mathbf{r}$ is close to $\mathbf{e_2}$, i.e., $\mathbf{e_1} + \mathbf{r} \approx \mathbf{e_2}$ mathematically. The score function is described as

$$f(t) = -\|\mathbf{e}_1 + \mathbf{r} - \mathbf{e}_2\|_2^2.$$

Similarly, we obtain entity embeddings $\overrightarrow{\mathbf{e_1}}$, $\overrightarrow{\mathbf{e_2}}$ and $\overleftarrow{\mathbf{e_1}}$, $\overleftarrow{\mathbf{e_2}}$ from the output of forward and reverse LSTMs, respectively, and require that $\overrightarrow{\mathbf{e_1}} + \mathbf{r} \approx \overrightarrow{\mathbf{e_2}}$ and $\overleftarrow{\mathbf{e_1}} + \mathbf{r} \approx \overleftarrow{\mathbf{e_2}}$. Hence, the score functions, respectively, are

$$\overrightarrow{f}(t) = -\left\|\overrightarrow{\mathbf{e_1}} + \mathbf{r} - \overrightarrow{\mathbf{e_2}}\right\|_2^2,$$
$$\overleftarrow{f}(t) = -\left\|\overleftarrow{\mathbf{e_1}} + \mathbf{r} - \overleftarrow{\mathbf{e_2}}\right\|_2^2.$$

4.3 Training of Joint Extractor

We mainly conduct two kinds of prediction on tag sequences and relations, in order to perform multi-triplets extraction.

To carry out tag sequence prediction on the basis of Bi-LSTM+CRF, we maximize the log-probability $p(\mathbf{y}|\mathbf{X})$ of the correct tag sequence,

$$\mathcal{L}_e = \log(p(\mathbf{y}|\mathbf{X})) = f(\mathbf{X}, \mathbf{y}) - \log(\sum_{y \in Y} e^{f(\mathbf{X}, \tilde{\mathbf{y}})}). \quad (1)$$

The purpose of maximizing $\mathcal{L}_e$, the loss from capturing entity features, is to encourage our model to construct a correct tag sequence.

Margin-based Relation Ranker. We accomplish the decision of relations between candidate entity pairs through ranking, i.e., appropriate relations will be ranked higher than the others. In order to better train our relation ranker, we construct a negative sample set $\mathcal{T}'$, which is composed of originally correct triplets with replaced relation.

Specifically for a triplet (e_1, e_2, r), we replace the original relation r by a random relation $r' \in \mathcal{R}$. The negative triplets $\mathcal{T}'$ can be described as

$$\mathcal{T}' = \{(e_1, e_2, r') | r' \in \mathcal{R}, r' \neq r\}.$$

To train relation embeddings and encourage discrimination between positive and negative triplets, we maximize the margin-based ranking loss function over training set in the embedding layer,

$$\mathcal{L}_{em} = \sum_{t \in \mathcal{T}} \sum_{t' \in \mathcal{T}'} \texttt{ReLu}(f(t') + \gamma - f(t)),$$

where $\gamma > 0$ is a hyperparameter to constrain the margin between positive and negative triplets, $\texttt{ReLu} = \max(0, x)$ (Glorot, Bordes, and Bengio 2011). Similarly, the loss functions of forward and reserve LSTM are described as

$$\overrightarrow{\mathcal{L}} = \sum_{t \in \mathcal{T}} \sum_{t' \in \mathcal{T}'} \texttt{ReLu}(\overrightarrow{f}(t') + \gamma - \overrightarrow{f}(t)),$$
$$\overleftarrow{\mathcal{L}} = \sum_{t \in \mathcal{T}} \sum_{t' \in \mathcal{T}'} \texttt{ReLu}(\overleftarrow{f}(t') + \gamma - \overleftarrow{f}(t)).$$

Hence, the loss from relation ranking is given by

$$\mathcal{L}_r = \mathcal{L}_{em} + \overrightarrow{\mathcal{L}} + \overleftarrow{\mathcal{L}}. \quad (2)$$

Combining $\mathcal{L}_e$ (Equation (1)) and $\mathcal{L}_r$ (Equation (2)), the final loss function is given by $\mathcal{L} = \mathcal{L}_e + \lambda \mathcal{L}_r$, where λ is a weighting hyperparameter to balance the two components. Then, the model is trained with stochastic gradient descent.

Multi-triplets Extraction. To conduct multi-triplets extraction, given a sentence, we firstly predict the tag sequence that obtains the maximum score by

$$\hat{y} = \arg\max_{\tilde{y} \in Y} f(\mathbf{X}, \tilde{y}).$$

Subsequently, we select as candidate entities the words of tag "R" in **RP** of TTS, which result in a set $\hat{\mathcal{E}} = \{\hat{e}_1, \ldots, \hat{e}_i, \ldots, \hat{e}_m\}$, where m is the number of candidates.

Afterwards, for candidate entity pair $(\hat{e}_i, \hat{e}_j)$, we generate the initial triplet set $\tilde{\mathcal{T}} = \{\tilde{t} = (\hat{e}_i, \hat{e}_j, r) | r \in \mathcal{R}\}$ and calculate the score by the function $f_c(\tilde{t}) = f(\tilde{t}) + \overrightarrow{f}(\tilde{t}) + \overleftarrow{f}(\tilde{t})$. For each entity pair, we choose only one triplet $\hat{t}$ such that

$$\hat{t} = \arg\max_{\tilde{t} \in \tilde{\mathcal{T}}} f_c(\tilde{t}).$$

If $f_c(\hat{t})$ is larger than a relation-specific threshold δ_r, $\hat{t}$ is a candidate triplet. The relation-specific threshold δ_r is determined by the accuracy on the validation set. Afterwards, we sort all candidate triplets as per $f_c(\hat{t})$, and the top-n triplets are considered as the extracted triplets, which are used to compare with the target triplets in test set. In each sentence, if and only if an extracted triplet matches the entities, their positions and the relation perfectly, the extraction is correct.

Discussion. The proposed joint extractor TME has a few notable advantages:

- Embedding translation mechanism directly retains position information of entities. For example, if we change the entities' position in (`Paris`, `France`, `Located in`), i.e., `France` + `Located in` $\approx$ `Paris`, which is considered as a negative triplet in our model. Compared with (Zheng et al. 2017b) that uses extra scheme to tag position information, TME is more adaptive.
- Compared with existing neural models for RC task, the relation ranker transforms the problem of detecting right relations into a ranking task, rather than classification. Hence, it partially resolves the hardness of learning features for class `other`, reducing the impact of unindented relations that ought to be `other`, and more importantly, makes it easy to train with negative samples.
- Compared with CoType (Ren et al. 2017), TME only uses the information within a sentence, which empowers the operability of the model in a variety of situations.

5 Experiments and Analysis

In this section, TME is evaluated against competing models, and we provide comprehensive analysis of the results.

Table 1: Dataset Statistics

Dataset	#Train	#Test	#Triplet	#Ent	#Rel
NYT-single	235,983	395	17,663	67,148	24
NYT-multi	63,602	1,000	17,494	25,894	24

5.1 Experiment Setup

Datasets. Experiments were carried out on two publicly available datasets NYT-single (Riedel, Yao, and McCallum 2010) and NYT-multi (statistics in Table 1). Specifically,

- **NYT-single** contains *New York Times* articles from 1987 to 2007, which in total includes 235k sentences. The invalid and duplicate sentences were filtered out and finally we obtained 67k sentences. In particular, the built-in test

Table 2: Experiment Results on NYT-single

Methods	Prec	Rec	F1
FCM	0.553	0.154	0.240
DS+logistic	0.258	0.393	0.311
LINE	0.335	0.329	0.332
MultiR	0.338	0.327	0.333
DS-Joint	0.574	0.256	0.354
CoType	0.423	0.511	0.463
NTS-Joint	**0.615**	0.414	0.495
TME (**top-1**)	0.583	0.485	**0.530**
TME (**top-2**)	0.515	0.508	0.511
TME (**top-3**)	0.458	**0.522**	0.489

Table 3: Experiment Results on NYT-multi

Methods	Prec	Rec	F1
CoType	0.385	0.340	0.361
NTS-Joint	0.533	0.336	0.412
TME-MR	0.638	0.421	0.507
TME-RR	0.423	0.452	0.437
TME-NS	0.558	0.496	0.525
TME (**top-1**)	**0.749**	0.436	0.551
TME (**top-2**)	0.696	0.478	**0.567**
TME (**top-3**)	0.631	**0.500**	0.558

set contains 395 sentences, most of which have single triplet in each sentence.

- **NYT-multi** is a derived dataset from NYT-single, which is specifically constructed for testing multi-triplets extraction. We randomly pulled out 1,000 sentences from the filtered NYT-single as test set, and took the rest as training set. Different from NYT-single, a large portion (39.1%) of the test set contains more than one triplet.

Competitors. For comparison, we employed the following models as baselines: DS+logistic (Mintz et al. 2009), MultiR (Hoffmann et al. 2011), DS-Joint (Li and Ji 2014), FCM (Gormley, Yu, and Dredze 2015), LINE (Tang et al. 2015), CoType (Ren et al. 2017), and NTS-Joint (Zheng et al. 2017b). In addition, we also made variants of TME for thorough investigation: (1) TME-RR: This variant uses random and stable relation embeddings $\mathbf{r}$ for model training; (2) TME-MR: This variant uses extra relation embeddings $\overrightarrow{\mathbf{r}}$ and $\overleftarrow{\mathbf{r}}$ to replace the the relation embeddings $\mathbf{r}$ in $\overrightarrow{f}(t)$ and $\overleftarrow{f}(t)$, respectively; and (3) TME-NS: This variant trains the model without negative sampling. Following NTS-Joint (Zheng et al. 2017b), we used precision (Prec), recall (Rec) and F-measure (F1) to evaluate the performance.

Implementations. For parameter setting, we selected the dimension of word embeddings d_w among {20, 50, 100, 200}, the dimension of character embeddings d_{ch} among {5, 10, 15, 25}, the dimension of capitalization embeddings d_c among {1, 2, 5, 10}, the margin γ between positive and negative triplets among {1, 2, 5, 10}, and the weighting hyperparameter λ among $\{0.2, 0.5, 1, 2, 5, 10, 20, 50\}$. The dropout ratio was set to 0 or 0.5. Stochastic gradient descent (Amari 1993) was called to optimize the loss function. We randomly chose 10% sentences in test set as validation set, and the rest was regarded as evaluation set. The optimal configurations were $\lambda = 10.0$, $\gamma = 2.0$, $d_w = 100$, $d_{ch} = 25$, $d_c = 5$, and $dropout = 0.5$.

The results of different extraction models on NYT-single are shown in Table 2, where k in **top-**k is used to limit the number of extracted triplets from each sentence. From Table 2, it reads that compared with other competitors, TME (**top-1**) achieves the state-of-the-art results, the F1 value is up to 0.530, and it outperforms the second runner, NTS-Joint, by 7%; besides, TME has evidently higher recall (increase of 17.1%) and negligibly lower precision (decrease of 5.4%), which proves that the ranking-based relation extractor handles relations between entity pairs more adaptively.

5.2 Results Analysis

Comparison Analysis. To prove the effectiveness of multi-triplets extraction, we use NYT-multi dataset and compare with some baselines. The results on NYT-multi are shown in Table 3 [3], and it reads that the F1 value in TME (**top-2**) is up to 0.567 and achieves a 36.7% improvement over NTS-Joint. Different from the results on NYT-single, the best results on NYT-multi are achieved by **top-2** rather than **top-1**, which can verify its abilities to process multi-triplets sentence. Besides, compared with TME-MR, TME-RR and TME-NS, TME also achieves better results, that is to say, (1) using the same relation embedding in different layers can effectively prevent embedding features from being offset after extracted by Bi-LSTM; and (2) negative sampling can improve the representation ability effectively.

Ablation Study. To show the effectiveness of each component, we remove one particular component at a time to understand its impact on the performance. Concretely, we investigated character embedding, CRF, TTS and single-layer translation-based model (by removing the score function at embedding layer, denoted -f, or Bi-LSTM layer, denoted -$\overrightarrow{f}$-$\overleftarrow{f}$); we also looked at the impact of pre-training and dropout mechanisms. Table 4 summarizes the results on NYT-multi. Compared with TME-f, multi-layer translation-based model gives the largest jump of 28.0% in F1 score, which verdicts the superiority of multi-layer model regarding triplet extraction. From the results of TME-TTS, we can conclude that **RP** and **TP** have positive effect on triplets extraction. Especially on **top-2**, the incorporation of **RP** brings a remarkable improvement (42.6%) in precision and negligible drop (-1.3%) in recall; this suggests that **RP** can effectively filter out entities irrelevant to target relations.

Parameter Analysis. In addition, we also analyze influence of different values of λ on performance, and the results are shown in Figure 4. If $\lambda > 20$ or $\lambda < 5$, the accuracy of F1 value declines. When $\lambda = 10$, TME achieves the balance between the performance of entity and relation extraction, and offers state-of-the-art F1 score.

[3] We could not reproduce some results of the prior work on NYT-multi, and only used the two having release source code, i.e., CoType and NTS-Joint, which were state-of-the-art.

Table 4: Ablation Study of TME on NYT-multi

Model	Top-1			Top-2			Top-3		
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
TME	**0.749**	**0.436**	**0.551**	**0.696**	0.478	**0.567**	**0.631**	0.500	**0.558**
-TTS (-**TP**)	0.741	0.436	0.549	0.680	0.478	0.561	0.610	0.498	0.548
-TTS (-**RP**)	0.610	0.376	0.465	0.488	**0.484**	0.486	0.400	**0.547**	0.462
-TTS (-**TP**-**RP**)	0.575	0.353	0.438	0.474	0.468	0.470	0.391	0.531	0.450
-Character	0.723	0.428	0.538	0.663	0.472	0.552	0.597	0.497	0.542
-CRF	0.690	0.414	0.517	0.608	0.470	0.530	0.522	0.495	0.509
- $\overrightarrow{f}$ - $\overleftarrow{f}$	0.552	0.310	0.398	0.521	0.368	0.431	0.468	0.399	0.431
- f	0.569	0.332	0.419	0.518	0.372	0.433	0.465	0.395	0.428
-Dropout	0.723	0.424	0.535	0.666	0.478	0.556	0.593	0.503	0.544
-Pretrain	0.686	0.411	0.514	0.613	0.466	0.530	0.539	0.495	0.516

Table 5: Case Study of TME (**Top-3**) on NYT-multi

Sentence I	... President **Jacques Chirac**[PER] of **France**[LOC] and Chancellor **Angela Merkel**[PER] of **Germany**[LOC] to press for agreement on a Security Council resolution demanding that *Iran*[LOC] stop ...		
Correct	(Jacques Chirac, nationality, France) (Angela Merkel, nationality, Germany)	**Predicted**	**(Jacques Chirac, nationality, France)** **(Angela Merkel, nationality, Germany)** (Jacques Chirac, nationality, Germany)
Sentence II	... grasping the critical need for the *United States*[LOC] to get **Afghanistan**[LOC] right, she moved to **Kandahar**[LOC] to help... Afghans for Civil Society, founded by the brother of **Hamid Karzai**[PER]...		
Correct	(Afghanistan, contains, Kandahar) (Hamid Karzai, place_of_birth, Kandahar) (Hamid Karzai, nationality, Afghanistan)	**Predicted**	(Kandahar, contains, Hamid Karzai) **(Afghanistan, contains, Kandahar)** **(Hamid Karzai, nationality, Afghanistan)**
Sentence III	... Across **Iraq**[LOC] , from **Mosul**[LOC] and **Ramadi**[LOC] to **Basra**[LOC] and **Kirkuk**[LOC], the lines of votes hummed with excitement, and with the hope that a permanent Iraqi government...		
Correct	(Iraq, contains, Mosul) (Iraq, contains, Ramadi) (Iraq, contains, Basra) (Iraq, contains, Kirkuk)	**Predicted**	**(Iraq, contains, Mosul)** **(Iraq, contains, Basra)** **(Iraq, contains, Ramadi)**

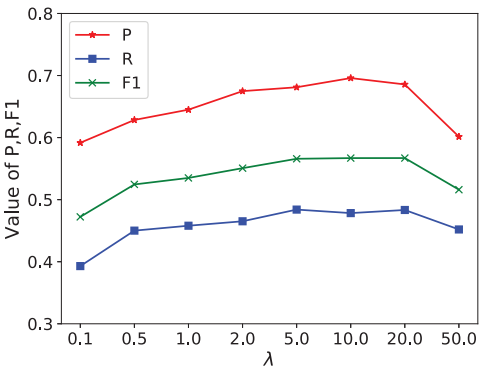

Figure 4: Performance of TME with Varying λ

Case Study. Table 5 shows the correct and predicted triplets of test set which can illustrate the performance of TME. In each sentence, the entities in bold denote the predicted entities with relationship and the Italic ones denote the predicted entities without relationship. In *Triplets* rows, The bold triplets represent the correct predicted triplets. Table 5 unveils that

- TME discovers multi-triplets in each sentence, not only when an entity is involved with different relations (cf. Sentence II), but also when the same relation appears in multiple entity pairs (cf. Sentence III).
- In Sentences I and II, the irrelevant entities `Iran` and `United States` are determined, which demonstrates that TTS helps improve the performance effectively.

6 Conclusion

In this paper, we identified the weakness of existing models that complex relationships between entities in sentences are overlooked, and candidate entity pairs are generated either with unpragmatic constraints or not carefully attended.

In our model TME, we devised a tri-part tagging scheme to recognize entities, and preclude irrelevant entities to target relations from participating relation extraction. Besides, TME employs an extra embedding to describe relation features, which enables a margin-based relation ranker trained with negative sampling strategy to decide appropriate relations between candidate entities.

Acknowledgments

This work was partially supported by NSFC under grants Nos. 61872446, 71690233 and 71331008, ARC DPs

170103710 and 180103411, and D2DCRC DC25002 and DC25003.

References

Amari, S. 1993. Backpropagation and stochastic gradient descent method. *Neurocomputing* 5(3):185–196.

Bollacker, K. D.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 1247–1250.

Chan, Y. S., and Roth, D. 2011. Exploiting syntactico-semantic structures for relation extraction. In *ACL*, 551–560.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. P. 2011. Natural language processing (almost) from scratch. *JMLR* 12:2493–2537.

Finkel, J. R.; Grenager, T.; and Manning, C. D. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 363–370.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *ICAIS*, 315–323.

Gormley, M. R.; Yu, M.; and Dredze, M. 2015. Improved relation extraction with feature-rich compositional embedding models. In *EMNLP*, 1774–1784.

Hammerton, J. 2003. Named entity recognition with long short-term memory. In *CoNLL*, 172–175.

Hoffmann, R.; Zhang, C.; Ling, X.; Zettlemoyer, L. S.; and Weld, D. S. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*, 541–550.

Kate, R. J., and Mooney, R. J. 2010. Joint entity and relation extraction using card-pyramid parsing. In *CoNLL*, 203–212.

Katiyar, A., and Cardie, C. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *ACL*, 917–928.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. In *NAACL*, 260–270.

Li, Q., and Ji, H. 2014. Incremental joint extraction of entity mentions and relations. In *ACL*, 402–412.

McCallum, A., and Li, W. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *HLT-NAACL*, 188–191.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *ACL*, 1003–1011.

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

Miwa, M., and Sasaki, Y. 2014. Modeling joint entity and relation extraction with table representation. In *EMNLP*, 1858–1869.

Passos, A.; Kumar, V.; and McCallum, A. 2014. Lexicon infused phrase embeddings for named entity resolution. In *CoNLL*, 78–86.

Ren, X.; Wu, Z.; He, W.; Qu, M.; Voss, C. R.; Ji, H.; Abdelzaher, T. F.; and Han, J. 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *WWW*, 1015–1024.

Riedel, S.; Yao, L.; and McCallum, A. 2010. Modeling relations and their mentions without labeled text. In *ECML-PKDD*, 148–163.

Rink, B., and Harabagiu, S. M. 2010. UTD: classifying semantic relations by combining lexical and semantic resources. In *ACL*, 256–259.

Roth, D., and Yih, W. 2004. A linear programming formulation for global inference in natural language tasks. In *CoNLL*, 1–8.

Santos, C. N. d.; Xiang, B.; and Zhou, B. 2015. Classifying relations by ranking with convolutional neural networks. In *ACL*, 626–634.

Shaalan, K. 2014. A survey of arabic named entity recognition and classification. *Computational Linguistics* 40(2):469–510.

Shen, Y., and Huang, X. 2016. Attention-based convolutional neural network for semantic relation extraction. In *COLING*, 2526–2536.

Singh, S.; Riedel, S.; Martin, B.; Zheng, J.; and McCallum, A. 2013. Joint inference of entities, relations, and coreference. In *CIKM*, 1–6.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. LINE: large-scale information network embedding. In *WWW*, 1067–1077.

Wang, L.; Cao, Z.; de Melo, G.; and Liu, Z. 2016. Relation classification via multi-level attention CNNs. In *ACL*.

Xu, Y.; Mou, L.; Li, G.; Chen, Y.; Peng, H.; and Jin, Z. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*, 1785–1794.

Yang, B., and Cardie, C. 2013. Joint inference for fine-grained opinion extraction. In *ACL*, 1640–1649.

Yu, X., and Lam, W. 2010. Jointly identifying entities and extracting relations in encyclopedia text via A graphical model approach. In *COLING*, 1399–1407.

Zhang, D., and Wang, D. 2015. Relation classification via recurrent neural network. *CoRR* abs/1508.01006.

Zhang, M.; Zhang, J.; and Su, J. 2006. Exploring syntactic features for relation extraction using a convolution tree kernel. In *NAACL*, 288–295.

Zheng, S.; Hao, Y.; Lu, D.; Bao, H.; Xu, J.; Hao, H.; and Xu, B. 2017a. Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing* 257:59–66.

Zheng, S.; Wang, F.; Bao, H.; Hao, Y.; Zhou, P.; and Xu, B. 2017b. Joint extraction of entities and relations based on a novel tagging scheme. In *ACL*, 1227–1236.