# Joint Extraction of Multiple Relations and Entities by Using a Hybrid Neural Network

Peng Zhou[1,2], Suncong Zheng[1,2], Jiaming Xu[1], Zhenyu Qi[1](✉), Hongyun Bao[1], and Bo Xu[1,2]

[1] Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
{zhoupeng2013,suncong.zheng,jiaming.xu
zhenyu.qi,hongyun.bao,xubo}@ia.ac.cn

**Abstract.** This paper proposes a novel end-to-end neural model to jointly extract entities and relations in a sentence. Unlike most existing approaches, the proposed model uses a hybrid neural network to automatically learn sentence features and does not rely on any Natural Language Processing (NLP) tools, such as dependency parser. Our model is further capable of modeling multiple relations and their corresponding entity pairs simultaneously. Experiments on the CoNLL04 dataset demonstrate that our model using only word embeddings as input features achieves state-of-the-art performance.

**Keywords:** Information extraction · Neural networks

## 1 Introduction

Entity and relation extraction is to detect entities and recognize their semantic relations from the given sentence. It plays a significant role in various NLP tasks, such as question answering [7] and knowledge base construction [16].

Traditional systems treat this task as a pipeline of two separated tasks, i.e., Named Entity Recognition (NER) [4] and Relation Classification (RC) [24]. Although adopting such a pipeline based method would make a system comparatively easy to assemble, it may encounter some limitations: First, the combination of these two components through a separate training way may hurt the performance. Consequently, errors in the upstream components (e.g., NER) are propagated to the downstream components (e.g., RC) without any feedback. Second, it over-simplifies the problem as multiple local classification steps without taking cross-task dependencies into consideration.

Recent studies show that joint modeling of entities and relations [9,12] is critical for achieving a high performance, since relations interact closely with entities. For instance, to recognize the triplet $\{\mathbf{Chapman}_{e1}, \mathbf{Kill}_{r}, \mathbf{Lennon}_{e2}\}$ in the following sentence:

*Lennon was murdered by Chapman outside the Dakota on Dec. 8, 1980.*

It may be useful to identify the relation *Kill* in this sentence, which constrains its arguments to be *Person* (or at least, not to be *Location*) and helps to enforce that **Lennon** and **Chapman** are likely to be *Person*, while **Dakota** is not.

However, most existing joint models are feature-based systems. They need complicated feature engineering and heavily rely on the supervised NLP toolkits, such as dependency parser, which might also lead to error propagation.

Recently, deep learning methods provide an effective way of reducing the number of handcrafted features. Miwa and Bansal [11] proposed an effective Recurrent Neural Networks (RNN) model that requires little feature engineering to detect entities first and then combines these two entities to detect relations. However, the sentence may contain lots of entities, and these entities will form too many entity pairs, as each two entities can form an entity pair. Normally, the number of relations is less than the number of entities in the sentence.[1] If the relations are detected first and used to recognize entity pairs, this will not only reduce the computational complexity but also extract triplets more exactly.

RNN also has disadvantages. Despite its ability to account for word order and long distance dependencies of an input sentence, RNN suffers from the problem that the later words make more influence on the final sentence representation than the former ones, ignoring the fact that important words can appear anywhere in the sentence. Though Convolutional Neural Networks (CNN) can relieve this problem by giving largely uniform importance to each word in the sentence, the long range dependency information in the sentence would be lost.

Most state-of-the-art systems [24] treat relation classification as a multi-class classification problem and predict one most likely relation for an input sentence. However, one sentence may contain multiple relations, and it is helpful to identify entity pairs by providing every possible relation.

Based on the analysis above, this paper presents a novel end-to-end model, dubbed BLSTM-RE, to jointly extract entities and relations. Firstly, Bidirectional Long Short-Term Memory Networks (BLSTM) is utilized to capture long-term dependencies and obtain the whole representation of an input sentence. Secondly, CNN is used to obtain a high level feature vector, which will be given to a sigmoid classifier. In this way, one or more relations can be generated. Finally, the whole sentence representation generated by BLSTM and the relation vectors generated by the sigmoid classifier are concatenated and fed to another Long Short-Term Memory Networks (LSTM) to predict entities. Our contributions are described as follows:

- This paper presents a novel end-to-end model BLSTM-RE to combine the extraction of entity and relation. It employs BLSTM and CNN to automatically learn features of the input sentence without using any NLP tools such as dependency parser. Therefore, it is simpler and more flexible.
- BLSTM-RE can generate one or more relations for an input sentence. Therefore it is capable of modeling multiple relations and their corresponding entity pairs simultaneously.

[1] The above example contains one relation and three entities, and these entities will form three entity pairs (or six entity pairs if the direction of relation is considered).

– Experimental results on the CoNLL04 dataset show that BLSTM-RE achieves better performance compared to the state-of-the-art systems.

## 2 Related Works

The task we address in this work is to extract triplets that are composed of two entities and the relation between these two entities. Over the years, a lot of models have been proposed, and these models can be roughly divided into two categories: the pipeline based method and the end-to-end based method. The former treats this task as a pipeline of two separated tasks, i.e., NER and RC, while the latter jointly models entities and relations.

### 2.1 Named Entity Recognition

NER, as a classical NLP task, has drawn research attention for a few decades. Most existing NER models are linear statistical models which include Conditional Random Fields (CRF) [21], and their performances rely on hand-crafted features extracted by NLP tools and external knowledge resources.

Recently, several neural network based models have been successfully applied to NER. Huang et al. [4] first proposed LSTM stacked with a CRF for sequential tagging tasks, including tagging Part Of Speech (POS), chunking and NER tasks, and produced state-of-the-art (or close to) accuracies. Lample et al. [8] applied character and word embeddings in LSTM-CRF and generated good results on NER for four languages.

### 2.2 Relation Classification

As to relation classification, besides traditional feature-based [18] and kernel-based approaches [23], several neural models have been proposed, including CNN and RNN. Zeng et al. [24] utilized CNN to extract lexical and sentence level features for relation classification; Zhang et al. [25] employed RNN to learn temporal features, long range dependency between nominal pairs. Vu et al. [19] combined CNN and RNN using a voting process to improve the results of RC.

This paper also implements a pipeline based model. It utilizes BLSTM to obtain the representation of a sentence, and then concatenates relation vectors, which are generated by the pre-trained relation classification model, just like CR-CNN proposed by Santos et al. [14], to extract entity pairs from the sentence.

### 2.3 Joint Extraction of Entities and Relations

As to end-to-end extraction of relations and entities, most existing models are feature-based systems, which include integer linear programming [20], card-pyramid parsing [6], global probabilistic graphical systems [17] and structured prediction [9,12]. Such models rely on handcrafted features extracted from NLP tools, such as POS. However, designing features manually is time-consuming, and using NLP tools may result in the increase of computational and additional propagation errors. Recently, deep learning methods provide an effective way of reducing the number of handcrafted features.

To reduce the manual work in feature extraction, three neural network based models have been proposed. Gupta et al. [1] utilized a unified multi-task RNN to jointly model entity recognition and relation classification tasks with a table representation. It needs to label $n(n+1)/2$ cells for a sentence of length $n$, while BLSTM-RE only needs to predict $m_r(n+1)$ tags, which are $m_r$ different relations, $n$ entity tags and one relation type, and $m_r$ is less than the number of relations in the sentence. Miwa et al. [11] utilized both bidirectional sequential and bidirectional tree-structured RNN to jointly extract entities and relations in a single model, which depended on a well-performing dependency parser. BLSTM-RE does not rely on the dependency parser, so it is more straightforward and flexible. Zheng et al. [26] proposed a hybrid neural network model to extract entities and relations. However they only joined the loss of NER and RC without considering the interactions between them, which may still hurt the performance.

To verify the effect of the sigmoid classifier, this paper proposes another joint model BLSTM-R. Different from BLSTM-RE, BLSTM-R treats relation classification as a multi-class classification problem and employs a softmax classifier to conduct relation classification instead of using a sigmoid classifier.

## 3 Model

As shown in Fig. 1, BLSTM-RE consists of five components: Input Layer, Embedding Layer, BLSTM Layer, RC Module and NER Module. The details of different components will be described in the following sections.

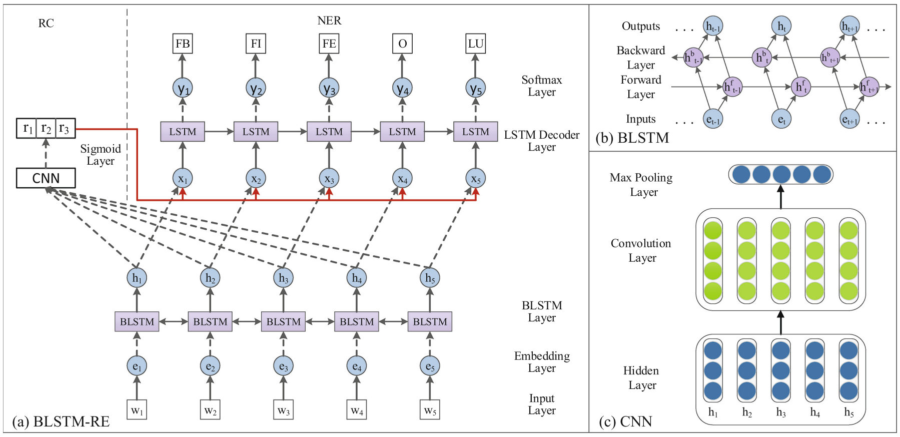
RC
NER
FB
FI
FE
O
LU
$y_1$ $y_2$ $y_3$ $y_4$ $y_5$
Softmax Layer
$r_1$ $r_2$ $r_3$
Sigmoid Layer
LSTM
LSTM Decoder Layer
CNN
$x_1$ $x_2$ $x_3$ $x_4$ $x_5$
$h_1$ $h_2$ $h_3$ $h_4$ $h_5$
BLSTM
BLSTM Layer
$e_1$ $e_2$ $e_3$ $e_4$ $e_5$
Embedding Layer
$w_1$ $w_2$ $w_3$ $w_4$ $w_5$
Input Layer
(a) BLSTM-RE
Outputs
Backward Layer
Forward Layer
Inputs
$h_{t-1}$ $h_t$ $h_{t+1}$
$h^b_{t-1}$ $h^b_t$ $h^b_{t+1}$
$h^f_{t-1}$ $h^f_t$ $h^f_{t+1}$
$e_{t-1}$ $e_t$ $e_{t+1}$
(b) BLSTM
Max Pooling Layer
Convolution Layer
Hidden Layer
$h_1$ $h_2$ $h_3$ $h_4$ $h_5$
(c) CNN

**Fig. 1.** An illustration of our model. (a): the overall architecture of BLSTM-RE, (b): BLSTM is utilized to capture sentence features, (c): CNN is utilized to capture a high level sentence representation. The dashed lines represent dropout.

## 3.1 Word Embeddings

If the input sentence consists of $l$ words $s = [w_1, w_2, \ldots, w_l]$, every word $w_i$ is converted into a real-valued vector $e_i$. For each word in $s$, we first look up the embedding matrix $W^{wrd} \in \mathbb{R}^{d \times |V|}$, where $V$ is a fixed-sized vocabulary and $d$ is the dimension of word embeddings. The matrix $W^{wrd}$ is a parameter to be learned, and $d$ is a hyper-parameter to be chosen by user. We transform a word $x_i$ into its word embeddings $e_i$ by using the matrix-vector product:

$$e_i = W^{wrd} v^i, \tag{1}$$

where $v^i$ is a one-hot vector of size $|V|$. Then the sentence is fed to the next layer as a real-valued matrix $emb_s = \{e_1, e_2, \ldots, e_l\} \in \mathbb{R}^{l \times d}$.

## 3.2 BLSTM Layer

LSTM [3] was proposed to overcome the gradient vanishing problem of RNN. The underlying idea is to introduce an adaptive gating mechanism, which decides the degree to which that LSTM units keep the previous state and memorize the extracted features of the current data input. From Embedding Layer, we obtain a real-valued matrix $emb_s = \{e_1, e_2, \ldots, e_l\}$, which will be processed by LSTM step by step. At time-step $t$, the memory $c_t$ and the hidden state $h_t$ are updated based on the following equations:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, e_t], \tag{2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t, \tag{3}$$

$$h_t = o_t \odot \tanh(c_t), \tag{4}$$

where $e_t$ is the input at the current time-step, $i_t$, $f_t$ and $o_t$ are the input gate, forget gate and output gate respectively, $\hat{c}$ is the current cell state, $\cdot$, $\sigma$ and $\odot$ denote dot product, the sigmoid function and element-wise multiplication respectively.

For the sequence modeling tasks, it is beneficial to have access to the past context as well as the future context. Schuster et al. [15] proposed BLSTM to extend the unidirectional LSTM by introducing a second hidden layer, where the hidden to hidden connections flow in the opposite temporal order. Therefore, BLSTM can exploit information from both the past and the future.

This paper also utilizes BLSTM to capture the past and the future information. As shown in Fig. 1(b), the network contains two sub-networks for the forward and backward sequence context respectively. The output of the $t^{th}$ word is shown in the following equation:

$$h_t = [\overrightarrow{h_t} \oplus \overleftarrow{h_t}]. \tag{5}$$

Here, the element-wise sum is used to combine the forward and backward pass outputs. In this paper, we set the hidden units of LSTM to the same size with word embeddings.

## 3.3 Relation Classification Module

As shown in Fig. 1, RC Module consists of three parts: Convolution Layer, Max Pooling Layer and Sigmoid Layer. The following sections will discuss each of the three layers in detail.

**Convolution Layer.** The hidden matrix $H = \{h_1, h_2, \ldots, h_l\} \in \mathbb{R}^{l \times d}$ is obtained from the BLSTM Layer, which contains the past and the future information of the input sentence $s$, and then is fed to the Convolution Layer. In this paper, one-dimensional narrow convolution [5] is utilized to extract higher level features of the sentence $s$. A convolution operation involves a filter $m \in \mathbb{R}^{k \times d}$, which is applied to a window of $k$ words to produce a new feature. For example, a feature $c_i$ is generated from a window of words $H_{i:i+k-1}$ by

$$c_i = f(m \cdot H_{i:i+k-1} + b), \tag{6}$$

here, $b \in \mathbb{R}$ is a bias term, and $f$ is a non-linear function such as hyperbolic tangent. This filter is applied to each possible window of words in the sentence $s$ to produce a feature map:

$$c = [c_1, c_2, \ldots, c_{l-k+1}]. \tag{7}$$

**Max Pooling Layer.** From Convolution Layer, we get a feature map $c \in \mathbb{R}^{l-k+1}$. Then, we employ the max-over-time pooling to select the maximum value of the feature map by

$$\hat{c} = max(c), \tag{8}$$

as the feature corresponding to the filter $m$. In RC Module, $n$ filters with different window sizes $k$ are utilized to learn complementary features. And the final vector $z$ is formed as:

$$z = [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_n]. \tag{9}$$

**Sigmoid Layer.** To find out whether a sentence contains multiple relations, we utilize a sigmoid classifier instead of a softmax classifier to classify the relations based on the feature $z$, which is defined as:

$$\hat{p}\left(y|s\right) = sigmoid\left(W_R \cdot z + b_R\right), \tag{10}$$

$$\hat{y} = \hat{p}\left(y|s\right) > \delta, \tag{11}$$

where $W_R \in \mathbb{R}^{n_r \times n}$, $n_r$ is the number of relations, $b_R \in \mathbb{R}$ is a bias term, and $\delta$ is a hyper-parameter to be chosen by user.

### 3.4 Named Entity Recognition Module

As shown in Fig. 1(a), NER module consists of two parts: LSTM Decoder Layer and Softmax Layer. Both of these two layers will be described in the following sections.

**LSTM Decoder Layer.** We treat entity detection as a sequential token tagging task and apply the *BIEOU* tagging scheme, where each tag means a token is the **B**egin, **I**nside, **E**nd, **O**utside and **U**nit of an entity mention respectively.

Note that relations are directed, and the same relation with opposite directions is considered to be two different classes. For example, compared to *Kill (Chapman, Lennon)*, *Kill (Lennon, Chapman)* expresses the opposite meaning that *Chapman* is murdered by *Lennon* in Sect. 1. This paper uses two different letters $F$ and $L$ to represent the former entity mention and the latter entity mention in the relation respectively. For example in Fig. 1(a), we assign $FB$, $FI$, $FE$ and $LU$ to two different entity mentions.

To extract entity pairs of different relations, we combine the relation vectors obtained by the RC Module to generate entity tags. If the sentence only contains one relation, at each time-step $t$, the output $h_t$ of the BLSTM Layer and the relation vector $r$ are concatenated and fed to the LSTM Decoder Layer.

$$y_t = LSTM(concat(h_t, r)), \tag{12}$$

here, $y_t \in \mathbb{R}^d$, $concat(h_t, r) \in \mathbb{R}^{d+n}$ represents $x_t$ in Fig. 1(a).

**Softmax Layer.** From the LSTM Decoder Layer, we get a real-valued matrix $O = \{y_1, y_2, \ldots, y_l\} \in \mathbb{R}^{l \times d}$. Then it is passed to the Softmax Layer to predict the named entity tags as follows:

$$\hat{p}\left(y|s\right) = softmax\left(W_T \cdot O + b_T\right), \tag{13}$$

$$\hat{y} = \arg\max_y \hat{p}\left(y|s\right), \tag{14}$$

where $W_T \in \mathbb{R}^{n_t \times d}$, $n_t$ is the number of entity tags, and $b_T \in \mathbb{R}$ is a bias term. In this way, we can get one relation and its possible entity pairs. If the sentence contains multiple relations, this process will be repeated several times, each time using a different relation vector.

## 4 Experimental Setups

In this section, we introduce the dataset, the evaluation metrics and the hyper-parameters used in this paper.

**Table 1.** Summary statistics of the dataset. Sent, Ment and Rel represent the number of sentences, entity mentions and relation instances respectively, L: average sentence length, M: maximum sentence length.

| Data | Sent | Ment | Rel | L | M |
|---|---|---|---|---|---|
| Train | 1,153 | 7,935 | 1,626 | 29.07 | 114 |
| Test | 288 | 2,025 | 422 | 28.94 | 118 |

### 4.1 Dataset

The primary experiments are conducted on a public dataset CoNLL04 [13][2]. The corpus defines four named entity types (*Location*, *Organization*, *Person* and *Other*) and five relation types (*Kill*, *Live_In*, *Located_In*, *OrgBased_In* and *Work_For*). Besides, it contains $1,441$ sentences that contain at least one relation. We randomly split these into training $(1,153)$ and test $(288)$, as same as Gupta et al. [1][3]. Summary statistics of the dataset are shown in Table 1.

### 4.2 Metric and Hyper-parameter Settings

We use the standard $F1$ measure to evaluate the performance of entity extraction and relation classification. An entity is considered correct if one of its tokens is tagged correctly. A relation for a word pair is considered correct if its relation type and its two entities are both correct.

We update the model parameters including weights, biases, and word embeddings using gradient based optimizer AdaDelta [22] to minimize binary cross-entropy loss for relation classification and cross-entropy loss for entity detection. As there is no standard development set, we randomly select 20% of the training data as the development set to tune the hyper-parameters. The final hyper-parameters are as follows.

The word embeddings are pre-trained by Miklov et al. [10], which are 300-dimensional. The number of hidden units of LSTM is 300. We use 300 convolution filters each for the window size of 3. We set the mini-batch size as 10 and the learning rate of AdaDelta as the default value 1.0. We set the threshold $\delta$ of the sigmoid classifier to 0.5, which is selected from $\{0.1, 0.2, \cdots, 0.9\}$ based on the performance of the development set. To alleviate overfitting, we use Dropout [2] on Embedding Layer, BLSTM Layer and Convolution Layer with a dropout rate of 0.3, 0.2 and 0.2 respectively. We also utilize $l2$ penalty with coefficient $1e^{-5}$ over the parameters.

## 5 Overall Performance

As other systems did not show the result of joint extraction of entities and relations on the CoNLL04 dataset, we only compare our models with two state-

[2] conll04.corp at cogcomp.cs.illinois.edu/page/resource_view/43.

[3] https://github.com/pgcool/TF-MTRNN/tree/master/data/CoNLL04.

**Table 2.** Comparison with previous results.

| Model | Settings | P | R | F1 |
|---|---|---|---|---|
| TF | pipeline | .647 | .522 | .577 |
| | end-to-end | **.760** | .509 | .610 |
| TF-MT | pipeline | .641 | .545 | .589 |
| | end-to-end | .646 | .531 | .583 |
| Ours | pipeline | .643 | .390 | .485 |
| | BLSTM-R | .691 | .481 | .567 |
| | BLSTM-RE | .747 | **.548** | **.632** |

**Table 3.** Comparison of our models on the task of entity detection.

| Model | P | R | F1 |
|---|---|---|---|
| pipeline | .597 | .410 | .486 |
| BLSTM-R | .779 | .648 | .708 |
| BLSTM-RE | .883 | .652 | .750 |

**Table 4.** Comparision for relation classification on the CoNLL04 dataset.

| | [Kate & Mooney] | | | [Miwa & Sasaki] | | | [Gupta & Schutze] | | | LSTM-RE | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| OrgBase_In | .662 | .641 | .647 | .768 | .572 | .654 | **.831** | .562 | .671 | .761 | **.783** | **.771** |
| Live_In | .664 | .601 | .629 | **.819** | .532 | .644 | .727 | .640 | .681 | .797 | **.739** | **.767** |
| Kill | .775 | .815 | .790 | .933 | .797 | .858 | .857 | **.894** | .875 | **.952** | .870 | **.909** |
| Located_In | .539 | .557 | .513 | .821 | .549 | .654 | **.867** | .553 | .675 | .804 | **.732** | **.766** |
| Work_For | .720 | .423 | .531 | .886 | .642 | .743 | **.945** | .671 | .785 | .845 | **.790** | **.817** |
| **Average** | .672 | .607 | .622 | **.845** | .618 | .710 | .825 | .664 | .737 | .832 | **.783** | **.806** |

of-the-art systems TF [12] and TF-MT [1]. Both TF and TF-MT mapped the entity and relation extraction task to a simple table-filling problem. And the table filling method needs to label $n(n+1)/2$ cells for a sentence of length $n$, while our models only need to predict $m_r(n+1)$ tags, where $m_r$ is much less than $n/2$. BLSTM-RE boosts the $F1$ score by 2.2%. Compared with these two models, our model BLSTM-RE is simpler and more effective.

Table 2 also indicates that both BLSTM-RE and BLSTM-R perform better than the pipeline model, mainly because that the pipeline model trains entities and relations separately without considering the interaction between them, while BLSTM-R and BLSTM-RE learn entities and relations simultaneously.

BLSTM-RE achieves better results than BLSTM-R, the reason is that the input sentence may contain many relations as shown in Table 1. The softmax classifier only generates one most likely relation, while the sigmoid classifier can generate several relations at a time. In this situation, BLSTM-R only models one triplet, while BLSTM-RE can model multiple triplets simultaneously.

## 5.1 Analysis of NER and RC

This section summarizes the performance of NER and RC individually, which means that an entity is considered correct if one of its tokens is tagged correctly and a relation is considered correct if its relation type is correct. Because these three systems [1,6,12] assumed that the entity boundaries were given and only recognized entity types, while we only recognize entity boundaries. Therefore, we only compare the effect of RC with them as shown in Table 4.

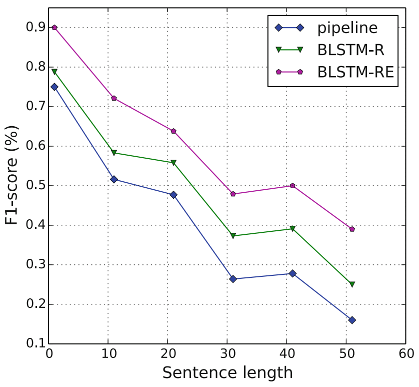

**Fig. 2.** Results *vs.* sentence length.

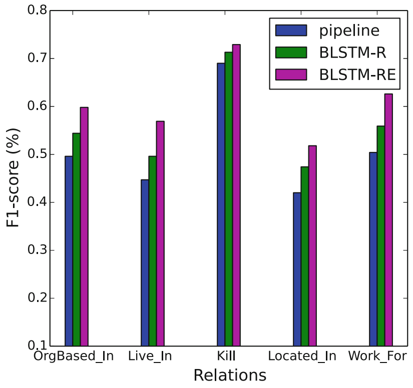

**Fig. 3.** Results *vs.* relations.

Table 3 shows the results of our three models on the task of NER. BLSTM-R and BLSTM-RE both have better performance than the pipeline model, which means that relation vectors are useful for the extraction of entities. Furthermore, BLSTM-RE is better than BLSTM-R, which shows that multi-label classification can effectively recognize relations than multi-class classification in this work.

The first two works [6,12] performed 5-fold cross-validation on the complete corpus. However, the folds were not available. We follow Gupta et al. [1] and report results on the same dataset. Since the standard divisions of the corpus are not the same, we cannot directly compare the results with the first two works [6,12]. But compared with TF-MT [1], BLSTM-RE shows an improvement of 6.9% in average $F1$ score.

## 5.2 Effect of the Sentence Length

Figure 2 depicts the performance of our models on sentences of different length. The $x$-axis and the $y$-axis represent sentence length and $F1$ score respectively. The sentences collected in the test set are no longer than 60 words. The $F1$ score is the average value of the sentences with length in the window $[n, n+9]$, where $n = \{1, 11, \ldots, 51\}$. Each data point is a mean score over five runs.

BLSTM-RE outperforms the other two models, and this suggests that learning multiple relations and entity pairs corresponding to the relations simultaneously can effectively extract triplets from sentences. At the same time, it shows that the $F1$ score declines with the length of sentence increasing. In the future work, we would like to investigate neural mechanisms to preserve long distance dependencies of sentences.

### 5.3 Effect of the Relations

Figure 3 depicts the performance of our models on different relations. The $x$-axis and $y$-axis represents relations and $F1$ score respectively. As the same as in Fig. 2, each data point is a mean score over five runs.

Different from Table 4, the $F1$ score involves entity and relation, which means that a relation is marked correct if the named entity boundaries and relation type are both correct. The figure shows that our models have different performance on different relations, and BLSTM-RE performs better than the other two models. All models perform better on relation *Kill* than the other four relations. There may be two main reasons. One is that all test sentences containing relation *Kill* do not contain other relations, and most of them contain keywords such as "kill", "death", and "assassinate", and therefore most of these sentences can be classified correctly when extracting relations. The other is that more than 80% of test sentences containing relation *Kill* only contain two entities. Thus, it is easy to recognize these entities.

## 6 Conclusions

This paper presents a novel end-to-end model BLSTM-RE to extract entities and relations. This model exploits BLSTM and CNN to automatically learn features from word embeddings without using any NLP tools. Thus, it is more straightforward and flexible. It treats relation classification as a multi-label classification problem and utilizes a sigmoid classifier to generate one or more relations. Therefore, it can model multiple relations and entity pairs at the same time. The effectiveness of BLSTM-RE is demonstrated by evaluating the model on the CoNLL04 dataset, and our model performs better than the pipeline based models and other end-to-end models. The experiment results also show that relation vectors obtained by RC Module are useful for the extraction of entities.

**Acknowledgments.** This research was supported by the National High Technology Research and Development Program of China (No. 2015AA015402) and the National Natural Science Foundation of China (No. 61602479). We thank the anonymous reviewers for their insightful comments.

## References

1. Gupta, P., Schutze, H., Andrassy, B.: Table filling multi-task recurrent neural network for joint entity and relation extraction. In: COLING (2016)
2. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. Comput. Sci. **3**(4), 212–223 (2012)
3. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
4. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. Comput. Sci. (2015)

5. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. In: ACL (2014)
6. Kate, R.J., Mooney, R.J.: Joint entity and relation extraction using card-pyramid parsing. In: ACL (2010)
7. Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., Socher, R.: Ask me anything: dynamic memory networks for natural language processing. Comput. Sci. (2016)
8. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: NAACL-HLT, pp. 260–270 (2016)
9. Li, Q., Ji, H.: Incremental joint extraction of entity mentions and relations. In: ACL, pp. 402–412 (2014)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
11. Miwa, M., Bansal, M.: End-to-end relation extraction using LSTMs on sequences and tree structures. In: ACL, pp. 1105–1116 (2016)
12. Miwa, M., Sasaki, Y.: Modeling joint entity and relation extraction with table representation. In: EMNLP, pp. 944–948 (2014)
13. Roth, D., Yih, W.: A linear programming formulation for global inference in natural language tasks. Technical report, DTIC Document (2004)
14. Santos, C.N.D., Xiang, B., Zhou, B.: Classifying relations by ranking with convolutional neural networks. Comput. Sci. (2015)
15. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
16. Shin, J., Wu, S., Wang, F., De Sa, C., Zhang, C., Ré, C.: Incremental knowledge base construction using deepdive. VLDB Endowm. **8**(11), 1310–1321 (2015)
17. Singh, S., Riedel, S., Martin, B., Zheng, J., Mccallum, A.: Joint inference of entities, relations, and coreference. In: The Workshop on Automated Knowledge Base Construction, pp. 1–6 (2013)
18. Suchanek, F.M., Ifrim, G., Weikum, G.: Combining linguistic and statistical analysis to extract relations from web documents. In: SIGKDD, pp. 712–717 (2006)
19. Vu, N.T., Adel, H., Gupta, P., et al.: Combining recurrent and convolutional neural networks for relation classification. In: NAACL-HLT, pp. 534–539 (2016)
20. Yang, B., Cardie, C.: Joint inference for fine-grained opinion extraction. In: ACL, pp. 1640–1649 (2013)
21. Yao, L., Sun, C., Li, S., Wang, X., Wang, X.: CRF-based active learning for Chinese named entity recognition. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 1557–1561 (2009)
22. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. Comput. Sci. (2012)
23. Zelenko, D., Aone, C., Richardella, A.: Kernel methods for relation extraction. J. Mach. Learn. Res. **3**(3), 1083–1106 (2010)
24. Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J., et al.: Relation classification via convolutional deep neural network. In: COLING, pp. 2335–2344 (2014)
25. Zhang, D., Wang, D.: Relation classification via recurrent neural network. Comput. Sci. (2015)
26. Zheng, S., Hao, Y., Lu, D., Bao, H., Xu, J., Hao, H., Xu, B.: Joint entity and relation extraction based on a hybrid neural network. Neurocomputing (2017)