# Joint Extraction of Entities and Relations Based on a Novel Graph Scheme

**Shaolei Wang**$^1$, **Yue Zhang**$^{*2}$, **Wanxiang Che**$^1$, **Ting Liu**$^1$
$^1$ Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China
$^2$ Singapore University of Technology and Design
{slwang, car, tliu}@ir.hit.edu.cn, yue_zhang@sutd.edu.sg

## Abstract

Both entity and relation extraction can benefit from being performed jointly, allowing each task to correct the errors of the other. Most existing neural joint methods extract entities and relations separately and achieve joint learning through parameter sharing, leading to a drawback that information between output entities and relations cannot be fully exploited. In this paper, we convert the joint task into a directed graph by designing a novel graph scheme and propose a transition-based approach to generate the directed graph incrementally, which can achieve joint learning through joint decoding. Our method can model underlying dependencies not only between entities and relations, but also between relations. Experiments on NewYork Times (NYT) corpora show that our approach outperforms the state-of-the-art methods.

## 1 Introduction

Extraction of entities and relations is a fundamental task of information extraction (IE). An example is shown in Figure 1, where the input is unstructured texts and the output includes entities and their semantic relations. There are strong connections between entities and relations, and also between relation labels in a sentence. For example, a *Live_In* relation suggests *Person* and *Location* entities, and vice versa. The *Live_In* relation (between "*John*" and "*California*") can be inferred from the *Live_In* (between "*John*" and "*Los Angeles*") and *Loc_In* (between "*Los Angeles*" and "*California*") relations.

The task has been traditionally solved as a pipeline of two separate sub-tasks: entity recognition [Nadeau and Sekine, 2007] and relation extraction [Zhou *et al.*, 2005]. This separation neglects the relevance between these two sub-tasks. Joint extraction of entities and relations can integrate information of entities and relations, and has achieved better results on this task. Joint models have been investigated using both statistical methods [Ren *et al.*, 2017; Miwa and Sasaki, 2014; Li and Ji, 2014] and neural methods [Zheng *et al.*, 2017; Katiyar and Cardie, 2017; Miwa and Bansal, 2016]. The performances of statistical models [Miwa and Sasaki, 2014;

*Email corresponding.

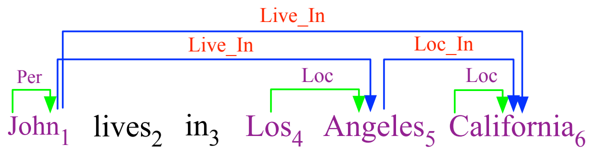
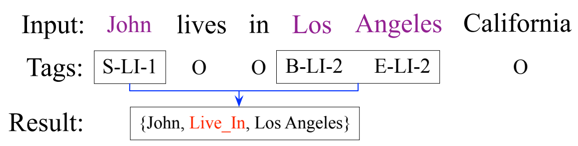

Live_In
Live_In
Loc_In
Per
Loc
Loc
John$_1$ lives$_2$ in$_3$ Los$_4$ Angeles$_5$ California$_6$

Figure 1: Entity and relation extraction (Green indicates entity arcs and blue indicates relation arcs.)

Li and Ji, 2014] heavily rely on complicated feature engineering and it is difficult to exploit global features.

Neural methods, in contrast, automatically learn non-local features by exploiting recurrent neural networks for learning sentence-level representations and have given the state-of-the-art results. However, most existing neural models [Katiyar and Cardie, 2017; Miwa and Bansal, 2016] extract entities and relations separately, achieving joint learning only through parameter sharing, but not joint decoding. This leads to a drawback that information between output entities and relations cannot be fully exploited, since no explicit features are used to model *output-output* dependencies. Zheng *et al.* [2017] is the only exception, designing a novel tagging scheme and converting the joint extraction task to a tagging problem. In their joint model, information of entities and relations is integrated into a unified tagging scheme and can be fully exploited. However, due to the transformation into a tagging task, the method only *indirectly* captures output structural correspondences, and is incapable of identifying overlapping relations (e.g. one entity can only have at most one relation).

To address this issue, we convert the joint task into a directed graph by designing a novel graph scheme, solved using a transition-based parsing framework [Zhang and Clark, 2011]. Different from traditional parsing tasks, nodes in our output structures may have multiple or no heads, as shown in Figure 1. We propose a novel transition system, which is a variant of the list-based arc-eager algorithm for non-projective tree parsing [Choi and McCallum, 2013]. By incrementally integrating entity information and their corresponding relation information, our method can model underlying dependencies not only between entities and relations, but also between relations. One challenge for designing a transition-based parsing system is the representation of parsing states (i.e. configurations), based on which the transition actions are disambiguated. We borrow the idea of neural parsing [Dyer *et al.*, 2015; Kiperwasser and Goldberg, 2016], designing a


Input: John lives in Los Angeles California
Tags: S-LI-1 O O B-LI-2 E-LI-2 O
Result: {John, Live_In, Los Angeles}
Figure 2: Example for the baseline tagging scheme.

special recursive neural network to model underlying *entity-relation* and *relation-relation* dependencies. We also use a Bi-LSTM to represent each sentence token to capture richer contextual information. The main contributions of this work are concluded as follows:

- We propose an intuitive graph scheme to jointly represent entities and relations, so that end to end relation extraction can be easily transformed into a parsing-like task
- Based on our graph scheme, we propose a novel transition system to generate the directed graph. In addition, we design a special recursive neural network to better model underlying *entity-relation* and *relation-relation* dependencies.
- We conduct our experiments on NewYork Times (NYT) corpora and the results show our method outperforms the state-of-the-art end-to-end methods.

The code is released[1].

## 2 Problem Definition

### 2.1 Baseline: Tagging Scheme

Zheng *et al.* [2017] treat the joint extraction task as a sequence labeling problem, proposing a novel tagging scheme. Figure 2 is an example of the tagging scheme. Tag "O" means that the corresponding word is independent of extracted entities and relations. In addition to "O", the other tags consist of three parts: the word position in the entity, the relation type, and the relation role. It uses the "BIES" (Begin, Inside, End, Single) signs to represent the position information of a word in the entity. The relation type information is obtained from a predefined set of relations. The relation role information is represented by the numbers "1" and "2", where "1" means that the word belongs to the first entity in the relation and "2" means that the word belongs to the second entity. As shown in Figure 2, "*Los*" is the first word of entity "*Los Angeles*" belonging to the second element of relation *Live_In*, so its tag is "B-LI-2". The other entity "*John*", which is the first element of relation *Live_In*, is labeled as "S-LI-1".

Based on this tagging scheme, Zheng *et al.* [2017] develop an end-to-end model with a biased loss function for the sequence labeling problem. State-of-the-art results are achieved thanks to the association between related entities in joint decoding. However, the method is incapable of identifying the overlapping relations. For instance, the sentence in Figure 1 contains three relations, in which every entity has two relations with other entities. But only one of the relations can be extracted under the tagging scheme.

[1]https://github.com/hitwsl/joint-entity-relation

| Transitions | Change of State |
|---|---|
| LEFT$_l$-REDUCE | $\frac{([\sigma\|i^*],\ \delta, e,\ [j^*\|\beta],\ R, E)}{(\sigma,\ \delta, e,\ [j^*\|\beta],\ R \cup \{(i^* \xleftarrow{l} j^*)\}, E)}$ |
| RIGHT$_l$-SHIFT | $\frac{([\sigma\|i^*],\ \delta, e,\ [j^*\|\beta],\ R, E)}{([\sigma\|i^*\|\delta\|j^*],\ [\,], e,\ \beta,\ R \cup \{(i^* \xrightarrow{l} j^*)\}, E)}$ |
| NO-SHIFT | $\frac{([\sigma\|i^*],\ \delta, e,\ [j^*\|\beta],\ R, E)}{([\sigma\|i^*\|\delta\|j^*],\ [\,], e,\ \beta,\ R, E)}$ |
| NO-REDUCE | $\frac{([\sigma\|i^*],\ \delta, e,\ [j^*\|\beta],\ R, E)}{(\sigma,\ \delta, e,\ [j^*\|\beta],\ R, E)}$ |
| LEFT$_l$-PASS | $\frac{([\sigma\|i^*],\ \delta, e,\ [j^*\|\beta],\ R, E)}{(\sigma,\ [i^*\|\delta], e,\ [j^*\|\beta],\ R \cup \{(i^* \xleftarrow{l} j^*)\}, E)}$ |
| RIGHT$_l$-PASS | $\frac{([\sigma\|i^*],\ \delta, e,\ [j^*\|\beta],\ R, E)}{(\sigma,\ [i^*\|\delta], e,\ [j^*\|\beta],\ R \cup \{(i^* \xrightarrow{l} j^*)\}, E)}$ |
| NO-PASS | $\frac{([\sigma\|i^*],\ \delta, e,\ [j^*\|\beta],\ R, E)}{(\sigma,\ [i^*\|\delta], e,\ [j^*\|\beta],\ R, E)}$ |
| O-DELETE | $\frac{([\sigma\|i^*],\ \delta, e,\ [j\|\beta],\ R, E)}{([\sigma\|i^*],\ \delta, e,\ \beta,\ R, E)}$ |
| GEN-SHIFT | $\frac{([\sigma\|i^*],\ \delta, e,\ [j\|\beta],\ R, E)}{([\sigma\|i^*],\ \delta,\ [j\|e], \beta,\ R, E)}$ |
| GEN-NER($y$) | $\frac{([\sigma\|i^*],\ \delta, [j\|e],\ [\beta],\ R, E)}{([\sigma\|i^*],\ \delta, [\,],\ [j^*\|\beta],\ R,\ E \cup \{j^*\})}$ |

Table 1: Transition actions, $*$ indicates an entity.

### 2.2 The Graph Scheme

Instead of label sequences, we transform entity mentions and their relations into a directed graph. The nodes in the graph correspond to words in the input sentence. The directed arcs are broadly categorized into: 1) entity arcs that represent internal structures of entities; 2) relation arcs that represent relations between entities, where head node means the first element of relation and modifier node means the second element of relation. To cope with the overlapping relations, the node in our directed graph can have multiple heads, which is different from traditional constituent parsing or dependency parsing graph. Figure 1 illustrates our graph representation, where the input sentence contains: 1) three entities, which are converted into corresponding green arcs with entity labels; and 2) three relations, which are converted into corresponding blue arcs with relation labels. Besides, the other words irrelevant to the final result have no corresponding arcs.

## 3 Method

### 3.1 Transition System

We propose a novel neural transition-based method, inspired by the list-based arc-eager algorithm [Choi and McCallum, 2013]. There are two types of transition actions: 1) entity actions, which are used to recognize entities; 2) relation actions, which are used to recognize relations between entities.

Formally, we use a tuple $(\sigma, \delta, e, \beta, R, E)$ to represent each state, where $\sigma$ is a stack holding processed entities, $\delta$ is a stack

| Transitions | Preconditions of transition actions |
|---|---|
| LEFT$_l$-$*$ | $[i \neq 0] \wedge \neg[(i \rightarrow^* j) \in R] \wedge (j \in E)$ |
| RIGHT$_l$-$*$ | $\neg[(j \rightarrow^* i) \in R] \wedge (j \in E)$ |
| $*$-REDUCE | $\neg[\exists k \in \beta.(i \rightarrow k) \vee (i \leftarrow k)] \wedge (j \in E)$ |
| $*$-SHIFT | $\neg[\exists k \in \sigma.(k \neq i) \wedge ((k \rightarrow j) \vee (k \leftarrow j))] \wedge (j \in E)$ |
| O-DELETE | $(j \notin E) \wedge (e = [\,])$ |
| GEN-SHIFT | $(j \notin E)$ |
| GEN-NER | $(j \notin E) \wedge (e \not\equiv [\,])$ |

Table 2: Preconditions of transition actions.

holding entities that are popped out of $\sigma$ but will be pushed back in the future, $e$ is a stack storing the partial entity chunk, and $\beta$ is a buffer holding unprocessed words. $R$ is a set of relation arcs. $E$ is a set of entity arcs. We use an index $i$ to represent word $w_i$ and entity $e_i$, respectively. $A$ is used to store the action history.

The set of actions is shown in Table 1. The first seven actions are used to generate relations, and the last three actions are used to generate entities. In particular, LEFT$_l$-REDUCE adds a relation arc with label $l$ from $e_j$ to $e_i$, and pops $e_i$ out of $\sigma$. RIGHT$_l$-SHIFT adds a relation arc with label $l$ from $e_i$ to $e_j$, and pushes all entities in $\delta$ and $e_j$ into $\sigma$. NO-SHIFT pushes all entities in $\delta$ and $e_j$ into $\sigma$. NO-REDUCE pops $e_i$ out of $\sigma$. LEFT$_l$-PASS adds a relation arc with label $l$ from $e_j$ to $e_i$, and moves $e_i$ to the front of $\delta$. RIGHT$_l$-PASS adds a relation arc with label $l$ from $e_i$ to $e_j$, and moves $e_i$ to the front of $\delta$. NO-PASS simply moves $e_i$ to the front of $\delta$. $(i^* \xrightarrow{l} j^*)$ is used to denote a relation arc from $e_i$ to $e_j$ with label $l$. $(i^* \rightarrow j^*)$ and $(i^* \rightarrow^* j^*)$ indicate that $e_i$ is a head and an ancestor of $e_j$ respectively. Note that all the relation actions are forbidden when the top element of $\beta$ is a word. O-DELETE pops $w_j$ out of $\beta$. GEN-SHIFT moves $w_j$ from $\beta$ to $e$. GEN-NER$(y)$ pops all items from the top of $e$ creating a "chunk", labels this with label y, pushes a representation of this chunk onto $\beta$, and an entity is added to $E$. All the entity actions are forbidden when the top element of $\beta$ is an entity.

Each action needs to satisfy certain preconditions to ensure the properties of a well-formed directed graph of entities and relations, as described in Table 2. To produce arcs pointing to entities with multiple heads, we design the preconditions of LEFT$_l$-$*$ and RIGHT$_l$-$*$ so that the dependency between a head and its modifier can be generated even if the modifier already has a head. Furthermore, we want to confirm that all heads and children of a word are found before the word is reduced. To this end, we set the head confirmation in the precondition of $*$-REDUCE to make sure no extra head of $e_i$ is in the buffer $\beta$.

Table 3 shows the sequence of state transitions given the sentence in Figure 1. The initial state is $([\ ], [\ ], [\ ], [1, \cdots, n], \emptyset, \emptyset)$, while the terminal state is $(\sigma, \delta, [\ ], [\ ], R, E)$. Transition actions are generated by consulting the gold-standard graph during training and a neural network classifier during decoding.

## 3.2 Search Algorithm

Based on the above transition system, our decoder searches for an optimal action sequence for a given sentence. The system is initialized by pushing all the input words and their representations onto $\beta$ in reverse order, such that the first word is at the top of $\beta$. $\sigma$, $\delta$, $e$ and $A$ each contains an empty-stack token. At each step, the system computes a composite representation of the model states (determined by the current configurations of $\beta$, $\sigma$, $\delta$, $e$ and $A$), which is used to predict an action to take. Decoding completes when $\beta$ and $e$ are both empty (except for the empty-stack symbol), regardless of the other states.

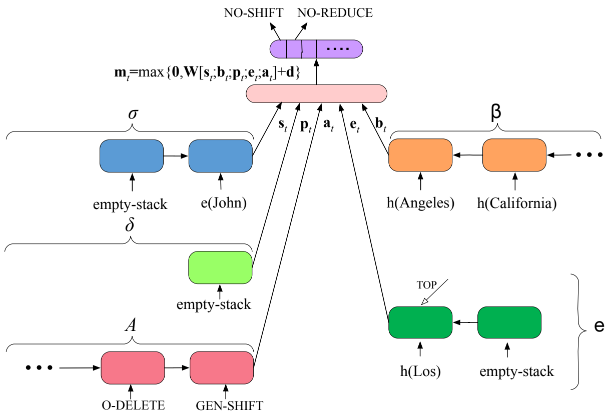


Figure 3: Representation of model state 6 in Table 3. $h(*)$ indicates the Bi-LSTM representation of each token, $e(*)$ indicates the composition of entities and their relations.

As shown in Figure 3, the model state representation at time $t$, which is written as $m_t$, is defined as:

$$m_t = \max\{0, W[s_t; b_t; p_t; e_t; a_t] + d\},$$

where $W$ is a learned parameter matrix, $s_t$ is the representation of $\sigma$, $b_t$ is the representation of $\beta$, $p_t$ is the representation of $\delta$, $e_t$ is the representation of $e$, $a_t$ is the representation of $A$, $d$ is a bias term. $(W[s_t; b_t; p_t; e_t; a_t] + d)$ is passed through a component-wise rectified linear unit (ReLU) for nonlinearity.

The model state $m_t$ is used to compute the probability of candidate actions at time $t$ as:

$$p(z_t|m_t) = \frac{\exp(g_{z_t}^{\mathrm{T}} m_t + q_{z_t})}{\sum_{z' \in \mathcal{A}(S,B)} \exp(g_{z'}^{\mathrm{T}} m_t + q_{z'})},$$

where $g_z$ is a column vector representing the embedding of the transition action $z$, and $q_z$ is a bias term for the action $z$. The set $\mathcal{A}(S, B)$ represents the set of valid actions that may be taken given the current state. Since $m_t$ encodes information about all previous decisions made by the transition system, the probability of any valid sequence of transition actions $z$ conditioned on the input can be written as:

$$p(z|w) = \prod_{t=1}^{|z|} p(z_t|m_t)$$

We then have

$$(E^*, R^*) = \mathrm{argmax}_{E,R} \prod_{t=1}^{|z|} p(z_t|m_t),$$

where $E^*$ is the best output entities, and $R^*$ is the best relations. Thus the extraction of entities and relations are merged in one transition-based system. To label a new input sequence at test time, the maximum probability action is chosen greedily until the algorithm reaches a termination state.

| **State** | **Transition** | $\sigma$ | $\delta$ | $e$ | $\beta$ | $R$ | $E$ |
|---|---|---|---|---|---|---|---|
| 0 | Initialization | $[\,]$ | $[\,]$ | $[\,]$ | $[1,\dots,6]$ | $\emptyset$ | |
| 1 | GEN-SHIFT | $[\,]$ | $[\,]$ | $[1]$ | $[2,\dots,6]$ | | |
| 2 | GEN-NER | $[\,]$ | $[\,]$ | $[\,]$ | $[1^*,\dots,6]$ | | $E \cup \{1 - \text{Per} \rightarrow 1\}$ |
| 3 | NO-SHIFT | $[1^*]$ | $[\,]$ | $[\,]$ | $[2,\dots,6]$ | | |
| 4 | O-DELETE | $[1^*]$ | $[\,]$ | $[\,]$ | $[3,\dots,6]$ | | |
| 5 | O-DELETE | $[1^*]$ | $[\,]$ | $[\,]$ | $[4,\dots,6]$ | | |
| 6 | GEN-SHIFT | $[1^*]$ | $[\,]$ | $[4]$ | $[5,6]$ | | |
| 7 | GEN-SHIFT | $[1^*]$ | $[\,]$ | $[4,5]$ | $[6]$ | | |
| 8 | GEN-NER | $[1^*]$ | $[\,]$ | $[\,]$ | $[5^*,6]$ | | $E \cup \{4 - \text{Loc} \rightarrow 5\}$ |
| 9 | RIGHT-SHIFT | $[1^*,5^*]$ | $[\,]$ | $[\,]$ | $[6]$ | $R \cup \{1 - \text{Live\_In} \rightarrow 5\}$ | |
| 10 | GEN-SHIFT | $[1^*,5^*]$ | $[\,]$ | $[6]$ | $[]$ | | |
| 11 | GEN-NER | $[1^*,5^*]$ | $[\,]$ | $[\,]$ | $[6^*]$ | | $E \cup \{6 - \text{Loc} \rightarrow 6\}$ |
| 12 | RIGHT-PASS | $[1^*]$ | $[5^*]$ | $[\,]$ | $[6^*]$ | $R \cup \{5 - \text{Loc\_In} \rightarrow 6\}$ | |
| 13 | RIGHT-SHIFT | $[1^*,5^*,6^*]$ | $[\,]$ | $[\,]$ | $[]$ | $R \cup \{1 - \text{Live\_In} \rightarrow 6\}$ | |

Table 3: Transition sequence for the entity and relation graph in Figure 1.

## 3.3 Input Representation

Two neural layers are used to represent the input, where the bottom layer is token embedding and the next layer is a Bi-LSTM layer to capture richer contextual information.

**Token Embedding** We use two vectors to represent each input token $t_i$: a learned word embedding $w_i$ and a fixed word embedding $\widetilde{w_i}$. The two vectors are concatenated, transformed by a matrix $V$ and fed to a rectified layer to learn a feature combination:

$$x_i = \max\{0, V[\widetilde{w}; w] + b\},$$

**Bi-LSTM Encoding** Given the token embeddings of an input sequence $x = (x_1, ..., x_n)$, bidirectional LSTM is used to process the sequence in both directions with two separate LSTM layers. The forward LSTM layer $\overrightarrow{h}$ encodes the input sequence from $x_1$ to $x_n$. In the similar way, the backward LSTM layer $\overleftarrow{h}$ will encode the input sequence from $x_n$ to $x_1$. We then concatenate $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ to represent word $t$'s encoding information, denoted as $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. Finally, the Bi-LSTM Embedding of the input sequence will be sent to the above structures of state representation.

## 3.4 State Representation

Shown in Figure 3, for better capturing non-local context information, we use stack LSTM [Dyer *et al.*, 2015] to represent different components of each state. For a conventional LSTM, new inputs are always added in the right-most position; but in a stack LSTM, the current location of a stack pointer determines which cell in the LSTM provides $c_{t-1}$ and $h_{t-1}$ when computing the new memory cell contents. In order to move the stack pointer from the right-most position, the stack LSTM provides a *pop* operation which moves the stack pointer to the previous element. Thus, the stack-LSTM can be understood as a stack implemented so that contents are never overwritten. By querying the output vector to which the stack pointer points, a continuous-space "summary" of the current stack configuration is available. As shown in Table 3, once a RIGHT-PASS action is taken (state 12), the *pop* operation will be taken to move the stack pointer from position of $5^*$ to $1^*$. Once a RIGHT-SHIFT action is taken (state 13), the elements in $\delta$ and top of $\beta$ will be added in the position that the stack pointer points ($1^*$) in order.

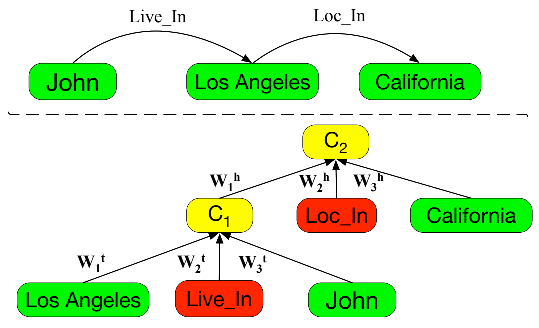


Figure 4: Relation representation of "*Los Angeles*", computed by recursively applying composition functions. $W^t$ is the parameters when "*Los Angeles*" is the modifier and $W^h$ is the parameters when "*Los Angeles*" is the head

## 3.5 Composition Functions

To model underlying *entity-relation* and *relation-relation* dependencies, our method incrementally integrates the entity information and corresponding relation information into the transition-based model.

**Entity Chunks** When GEN-NER($y$) is executed, the algorithm shifts the sequence of words on $e$ to the top of $\beta$ as a single completed chunk. To compute an embedding of this sequence, we run a bidirectional LSTM over the embeddings of its constituent words together with the chunk type (i.e., y). This function is denoted as $g(u, ..., v, r_y)$, where $r_y$ is a learned embedding of a label type. Thus, $\beta$ contains a single vector representation for each labeled entity chunk.

**Relation Labels** Recursive neural network models enable complex phrases to be represented compositionally in terms of their components and relations [Dyer *et al.*, 2015]. Given a directed relation arc, which points from a *head* node $h$ to a *modifier* node $m$, we combine both head-modifier pair and modifier-head pair, and use the combinations to update the embeddings of head node and modifier node separately. Formally, for the head-modifier pair, we have

$$c = \tanh(W^h[H; M; R] + e^h)$$

where $W^h$ is a learned parameter matrix, $H$ is neural embedding of the head entity, $M$ is neural embedding of the modifier entity, $R$ is vector embedding of the relation, $e^h$ is a bias term.

Similary, for the modifier-head pair, we have

$$c = \tanh(W^t[M; H; R] + e^t)$$

where $W^t$ is a learned parameter matrix, $e^t$ is a bias term, and the others are the same with head-modifier pair.

To simplify the parameterization of our composition function, we combine the pairs one at a time, building up more complicated structures in the order they are "reduced" in the model. Figure 4 shows an example when updating the entity "*Los Angeles*", where the *Live_In* relation is generated firstly.

## 4 Experiments

### 4.1 Experimental settings

**Dateset** To directly compare with Zheng *et al.* [2017], we use the public dataset NYT[2] as our main data set, which is produced by distant supervision [Ren *et al.*, 2017]. The training data set is obtained by means of distant supervision methods without manually labeling and contains 353k triplets in total. While the test set is manually labeled and contains 3, 880 triplets. Besides, the size of relation set is 24.

**Evaluation Metrics** We adopt standard Precision (Prec), Recall (Rec) and F1 score to evaluate the model. The labels of entity types are not considered when computing the final F1-score [Ren *et al.*, 2017; Zheng *et al.*, 2017]. In other words, a triplet is regarded as correct when its relation type and head offsets of the two corresponding entities are both correct. We follow Zheng *et al.* [2017], creating a validation set by randomly sampling 10% data from test set and use the remaining data as evaluation.

### 4.2 Hyperparameters and Training Details

Given a set of training data, the training goal is to maximize the likelihood of each gold action given the current model state. We update all model parameters by backpropagation using stochastic gradient descent (SGD) with a learning rate of 0.01 and gradient clipping at 5.0. We regularize our network using dropout with rate 0.3 tuned using the development set. Following Dyer *et al.* [2015], we use a variant of the skip n-gram model, namely structured skip n-gram [Ling *et al.*, 2015], to create word embeddings. The AFP portion of English Gigaword corpus (version 5) is used as the training corpus and the embedding dimension is 100. We have 2 hidden layers in our network and the dimensionality of the hidden units is 100.

### 4.3 Experimental Results

**Baselines** We compare our method with several state-of-the-art extraction methods, which can be divided into the following categories: the pipelined methods, the jointly extracting methods, and the end-to-end methods. For the pipelined methods, the NER results are obtained by Ren *et al.* [2017], then several classical relation classification methods are applied to detect the relations. These methods include: (1) **DS-logistic** [Mintz *et al.*, 2009] is a distant supervised and feature

[2]The dataset can be downloaded at: https://github.com/shanzhenren/CoType.

| Method | Prec. | Rec. | F1 |
|---|---|---|---|
| FCM [Gormley *et al.*, 2015] | 55.3 | 15.4 | 24.0 |
| DS+logistic [Mintz *et al.*, 2009] | 25.8 | 39.3 | 31.1 |
| LINE [Tang *et al.*, 2015] | 33.5 | 32.9 | 33.2 |
| MultiR [Hoffmann *et al.*, 2011] | 33.8 | 32.7 | 33.3 |
| DS-Joint [Li and Ji, 2014] | 57.4 | 25.6 | 35.4 |
| CoType [Ren *et al.*, 2017] | 42.3 | 51.1 | 46.3 |
| LSTM-LSTM-Bias | 61.5 | 41.4 | 49.5 |
| LSTM-LSTM-Bias* | 60.8 | 41.3 | 49.1 |
| **Our Method** | **64.3** | **42.1** | **50.9** |

Table 4: Comparison with previous state-of-the-art methods on NYT. The first part (from row 1 to row 3) is the pipelined methods, the second part (row 4 to 6) is the jointly extracting methods, and the third part (row 7 to 9) is the end-to-end methods.

| Method | Prec. | Rec. | F1 |
|---|---|---|---|
| ALL | 64.3 | 42.1 | **50.9** |
| -composition | 62.3 | 41.2 | 49.6 |
| -Bi-LSTM | 62.3 | 40.5 | 49.1 |

Table 5: Ablation test on NYT.

based method, which combines the advantages of supervised IE and unsupervised IE features; (2) **LINE** [Tang *et al.*, 2015] is a network embedding method, which is suitable for arbitrary types of information networks; (3) **FCM** [Gormley *et al.*, 2015] is a compositional model that combines lexicalized linguistic context and word embeddings for relation extraction.

The joint methods are listed as follows: (4) **DS-Joint** [Li and Ji, 2014] is a supervised method, which jointly extracts entities and relations using structured perceptron on human-annotated dataset; (5) **MultiR** [Hoffmann *et al.*, 2011] is a typical distant supervised method based on multi-instance learning algorithms to combat the noisy training data; (6) **CoType** [Ren *et al.*, 2017] is a domain independent framework by jointly embedding entity mentions, relation mentions, text features and type labels into meaningful representations.

**LSTM-LSTM-Bias** [Zheng *et al.*, 2017] is the baseline end-to-end method in §2, **LSTM-LSTM-Bias*** is our implementation.

**Results** Table 4 shows the results. Our transition-based method achieves significant improvements over all the baselines in F1 score. In particular, it achieves 4.6 point improvement over the best jointly extracting method [Ren *et al.*, 2017], and 1.4 point improvement over the best end-to-end sequence labeling method [Zheng *et al.*, 2017], demonstrating the effectiveness of our model on modeling and predicting entities and relations.

The joint methods by multi-task learning are better than pipelined methods, and the end-to-end methods are better than most of the joint methods. This result indicates the importance of joint decoding, which has stronger power of exploiting the dependencies between entities and relations, and also between relation labels in a sentence.

It is worth noting that the precision of our method is much higher compared to all the other methods. We attribute the success to the strong ability to model feature representations of entities and relations, and also the joint decoding.

| | |
|---|---|
| **Standard S1:** | CARVING THE GRANITE A Park as Eight Miles of Exits The Flume, [Cannon Mountain]$_{\text{LOC:CONTAIN-2}}$ and the Old Man of the Mountain Historic Site are all part of Franconia Notch State Park in [New Hampshire]$_{\text{LOC:CONTAIN-1}}$. |
| **LSTM-LSTM-Bias*:** | CARVING THE GRANITE A Park as Eight Miles of Exits The Flume, [Cannon Mountain] and the Old Man of the Mountain Historic Site are all part of Franconia Notch State Park in [New Hampshire]. |
| **Our Model:** | CARVING THE GRANITE A Park as Eight Miles of Exits The Flume, [Cannon Mountain]$_{\text{LOC:CONTAIN-2}}$ and the Old Man of the Mountain Historic Site are all part of Franconia Notch State Park in [New Hampshire]$_{\text{LOC:CONTAIN-1}}$. |
| **Standard S2:** | The US offered to locate the missile system in Poland, drawing furious objections from [Russia]$_{\text{LOC-LAC-1}}$, though [Washington]$_{\text{LOC-LAC-1}}$ argues that the system is built to defend against [Iran]$_{\text{LOC-LAC-2},\text{LOC-LAC-2}}$, principally. |
| **LSTM-LSTM-Bias*:** | The US offered to locate the missile system in Poland, drawing furious objections from [Russia]$_{\text{LOC-LAC-1}}$, though [Washington] argues that the system is built to defend against [Iran]$_{\text{LOC-LAC-2}}$, principally. |
| **Our Model:** | The US offered to locate the missile system in Poland, drawing furious objections from [Russia]$_{\text{LOC-LAC-1}}$, though [Washington]$_{\text{LOC-LAC-1}}$ argues that the system is built to defend against [Iran]$_{\text{LOC-LAC-2},\text{LOC-LAC-2}}$, principally. |

Table 6: Output from LSTM-LSTM-Bias and our model. The first row for each example is the gold standard. "*LOC*" is entity type, "*CONTAIN*" and "*LAC*" are relation types, "1" and "2" mean direction of relation. The color of "*LOC-LAC**" refers to relation instance.

**Ablation Tests** To demonstrate the effect of Bi-LSTM representation and relation composition function, we further conduct a set of ablation experiments. For the former, we directly send the token embedding of the input sentence to the above structures of state representation. For the latter, we only update each entity embedding by concatenating its original embedding with relation embedding once a relation arc is generated, ignoring the corresponding head or modifier entity. As shown in Table 5, the F1-score decreases heavily without each strategy, which indicates that it is very important to capture richer contextual information for token embedding and model feature representations of entities and relations.

**Case Study** We compare our method with LSTM-LSTM-Bias [Zheng *et al.*, 2017] on some cases, as shown in Table 6. As demonstrated by S1, when the distance between two interrelated entities is large, it is more difficult for the LSTM-LSTM-Bias method to identify their relation. However, thanks to the use of stack LSTM state representation, our transition-based method can capture more global feature representations, which make it more powerful identifying long-term relations.

Unlike the LSTM-LSTM-Bias method, our method can identify overlapping relations. S2 in Table 6 shows an example, which cannot be identified by the LSTM-LSTM-Bias method because of its model restriction. Our transition systems have the ability to handle multiple head or tail nodes, which make it suitable for such situation. In addition, our method directly models feature representations of entities and relations by using specially designed composition function, which makes it more powerful when dealing with overlapping relations.

## 5 Related Work

Two main methods have been proposed for entity mention and relation extraction, namely the pipeline method and the joint learning method. The former treats this task as two separated tasks, i.e., named entity recognition (NER) [Florian *et al.*, 2010; Kuru *et al.*, 2016] and relation extraction [Bunescu and Mooney, 2005; Liu *et al.*, 2015; Lin *et al.*, 2016].

Existing joint methods include feature-based statistical systems [Ren *et al.*, 2017; Miwa and Sasaki, 2014; Li and Ji, 2014], and neural models [Katiyar and Cardie, 2017; Miwa and Bansal, 2016; Zheng *et al.*, 2017]. Miwa and Bansal [2016] propose a neural method comprised of a sequence-based LSTM for entity identification and a separate tree-based dependency LSTM layer for relation classification. Their model depends critically on access to dependency trees and achieves joint learning only through parameter sharing. Katiyar and Cardie [2017] propose an attention-based joint neural model without accessing to dependency trees. This model extracts the entities and relations from left to right incrementally. However, it does not perform joint decoding and does not model the dependencies between different relations. Zheng *et al.* [2017] convert the joint task to a sequence labeling problem, achieving joint decoding for entities and relations in one task. Similar with Zheng *et al.* [2017], we build a neural model with joint decoding. Different from Zheng *et al.* [2017], however, we transform the joint task into a graph problem and propose a transition-based method.

There has been work using transition-based methods [Zhang and Clark, 2011; Nivre, 2008] to produce dependency trees and directed acyclic graphs (DAGs), but little work on more accurately directed graph in our joint tasks. Choi and McCallum [2013] use the list-based arc-eager algorithm for non-projective trees. Our model extends the method for yielding directed graph structure.

Recently, neural transition-based parsers have achieved highly competitive accuracies thanks to the modeling of *output-output* relations [Dyer *et al.*, 2015; Kiperwasser and Goldberg, 2016; Lample *et al.*, 2016; Liu and Zhang, 2017; Zhou *et al.*, 2015; Wang *et al.*, 2018]. We are inspired by these methods. To our knowledge, we are the first to investigate neural transition-based parsing framework for entity and relation extraction.

## 6 Conclusion

We proposed a neural transition-based approach for joint entity and relation extraction. Compared with existing neural methods, our method can model underlying dependencies not only between entities and relations, but also between relations. Experiments show that our model achieves the state-of-the-art F-scores on a standard NewYork Times (NYT) benchmark.

## Acknowledgments

We thank the anonymous reviewers for their valuable suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via grant 61632011 and 61772156, and BlueFire AI Ltd via Singapore Grant IGIPTET1702001 at Singaproe University and Technology and Design, through which the first author was funded as a visiting student to undertake this research. Yue Zhang is the corresponding author.

## References

[Bunescu and Mooney, 2005] Razvan C Bunescu and Raymond J Mooney. A shortest path dependency kernel for relation extraction. In *Proc. of EMNLP*, 2005.

[Choi and McCallum, 2013] Jinho D. Choi and Andrew McCallum. Transition-based dependency parsing with selectional branching. In *Proc. of ACL*, pages 1052–1062, 2013.

[Dyer *et al.*, 2015] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL and IJCNLP*, 2015.

[Florian *et al.*, 2010] Radu Florian, John F Pitrelli, Salim Roukos, and Imed Zitouni. Improving mention detection robustness to noisy input. In *Proc. of EMNLP*, 2010.

[Gormley *et al.*, 2015] Matthew R Gormley, Mo Yu, and Mark Dredze. Improved relation extraction with feature-rich compositional embedding models. *arXiv preprint arXiv:1505.02419*, 2015.

[Hoffmann *et al.*, 2011] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proc. of ACL*, pages 541–550, 2011.

[Katiyar and Cardie, 2017] Arzoo Katiyar and Claire Cardie. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proc. of ACL*, pages 917–928, July 2017.

[Kiperwasser and Goldberg, 2016] Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. *TACL*, 4:313–327, 2016.

[Kuru *et al.*, 2016] Onur Kuru, Ozan Arkan Can, and Deniz Yuret. Charner: Character-level named entity recognition. In *Proc. of COLING*, pages 911–921, December 2016.

[Lample *et al.*, 2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proc. of NAACL*, pages 260–270, June 2016.

[Li and Ji, 2014] Qi Li and Heng Ji. Incremental joint extraction of entity mentions and relations. In *ACL (1)*, 2014.

[Lin *et al.*, 2016] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proc. of ACL*, pages 2124–2133, August 2016.

[Ling *et al.*, 2015] Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. Two/too simple adaptations of word2vec for syntax problems. In *Proc. of NAACL*, 2015.

[Liu and Zhang, 2017] Jiangming Liu and Yue Zhang. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5, 2017.

[Liu *et al.*, 2015] Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. A dependency-based neural network for relation classification. In *Proc. of ACL*, pages 285–290, 2015.

[Mintz *et al.*, 2009] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proc. of ACL*, 2009.

[Miwa and Bansal, 2016] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. In *Proc. of ACL*, 2016.

[Miwa and Sasaki, 2014] Makoto Miwa and Yutaka Sasaki. Modeling joint entity and relation extraction with table representation. In *EMNLP*, pages 1858–1869, 2014.

[Nadeau and Sekine, 2007] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[Nivre, 2008] Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008.

[Ren *et al.*, 2017] Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proc. of the 26th International Conference on World Wide Web*, 2017.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proc. of the 24th International Conference on World Wide Web*, 2015.

[Wang *et al.*, 2018] Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. A neural transition-based approach for semantic dependency graph parsing. 2018.

[Zhang and Clark, 2011] Yue Zhang and Stephen Clark. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151, 2011.

[Zheng *et al.*, 2017] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. Joint extraction of entities and relations based on a novel tagging scheme. In *Proc. of ACL*, pages 1227–1236, July 2017.

[Zhou *et al.*, 2005] GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proc. of ACL*, pages 427–434, 2005.

[Zhou *et al.*, 2015] Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proc. of ACL*, pages 1213–1222, 2015.