Extracting Entities and Events as a Single Task Using a Transition-Based Neural Model

Junchi Zhang[1*], **Yanxia Qin**[2*], **Yue Zhang**[3,4*], **Mengchi Liu**[1†], **Donghong Ji**[5†]
[1]School of Computer, Wuhan University, China
[2]School of Computer Science and Technology, Donghua University, China
[3]School of Engineering, Westlake University, China
[4]Institute of Advanced Technology, Westlake Institute for Advanced Study, China
[5]School of Cyber Science and Engineering, Wuhan University, China
{zjc, mengchi, dhji}@whu.edu.cn, qolina@gmail.com, zhangyue@westlake.edu.cn

Abstract

The task of event extraction contains subtasks including detections for entity mentions, event triggers and argument roles. Traditional methods solve them as a pipeline, which does not make use of task correlation for their mutual benefits. There have been recent efforts towards building a joint model for all tasks. However, due to technical challenges, there has not been work predicting the joint output structure as a single task. We build a first model to this end using a neural transition-based framework, incrementally predicting complex joint structures in a state-transition process. Results on standard benchmarks show the benefits of the joint model, which gives the best result in the literature.

1 Introduction

There is a surge of interest in extracting structure information from plain text. Event extraction is an essential and challenging task, which has been shown beneficial to a wide range of downstream tasks, including question answering [Srihari and Li, 2000] and stock prediction [Ding *et al.*, 2014]. According to the ACE2005 dataset[1], event extraction contains three subtasks, namely, detecting entity mentions that represent an object or set of objects in the world, identifying event triggers, which are words in a sentence representing the predicates of an event, and identifying arguments, namely associating triggers with entities involved in the event.

Most prior work model event extraction by assuming that gold standard entities are provided [Ji and Grishman, 2008; Liao and Grishman, 2010; Hong *et al.*, 2011; Chen *et al.*, 2015; Nguyen *et al.*, 2016; Liu *et al.*, 2018b]. However, entity mention detection is a task strongly related to trigger identification and argument identification. Consider the sentence in Figure 1 for example, The expression *"10 years in jail"* can likely be divided into *"10 years"* and *"jail"* by naive NER systems. In contrast, by first recognizing that *"Hanh"* plays *Defendant* role in a *Sentence* event, a system can be more confident in identifying it as one piece, serving as one argument. These interdependencies cannot be captured by pipelined approaches, which perform the subtasks separately in a two-stage or three-stage procedure. Li *et al.*, [2013] show that by using automatically extracted entities in a pipelined approach, the performance of argument identification drops by at least 10% F-scores.

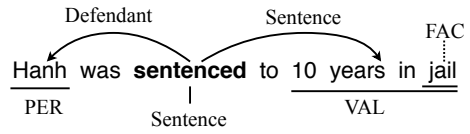

Figure 1: Example sentence from ACE05 dataset.

To address this issue, there have been efforts towards joint modeling of the three subtasks [Li *et al.*, 2014; Judea and Strube, 2016; Yang and Mitchell, 2016; Nguyen and Nguyen, 2019]. Yang and Mitchell [2016] consider structural dependencies among subtasks, by adopting a two-stage reranking procedure, first selecting the k-best output of event triggers and entity mentions, then performing joint inference via reranking. Very recently, Nguyen and Nguyen [2019] build a multi-task model that exploits mutual benefits among the three tasks, by sharing common encoding layers given an input sentence. In this setting, output structures of entity mentions, event triggers and argument semantic roles are decoded separately.

Both the methods of Yang and Mitchell [2016] and that of Nguyen and Nguyen [2019] can be regarded as efforts towards a fully joint event extractor. However, both methods still follow a pipeline framework by first predicting triggers and entities from texts, and then making assignments of arguments to triggers. Ideally, the trigger and argument structures in Figure 1 should be taken as one integrated graph, the structure of which is predicted without predicting trigger and entity span substructures in isolation, so that the maximum potential of information interaction can be exploited.

To this end, we make use of a transition-based framework [Nivre, 2008; Zhang and Clark, 2011; Dyer *et al.*,

*The first three authors contributed equally.
†Corresponding Authors
[1]https://catalog.ldc.upenn.edu/ldc2006t06

2015], which constructs a complex output structure holistically, through a state-transition process with incremental output-building actions. Transition-based methods have been applied to syntatic parsing [Dyer *et al.*, 2015], semantic parsing [Wang *et al.*, 2018b], entity recognition [Lample *et al.*, 2016], relation extraction [Wang *et al.*, 2018a] and many other NLP tasks, giving highly competitive accuracies. We design a novel transition system for joint event extraction, constructing the structure of Figure 1 incrementally from left to right, without differentiating subtask structures. In this process, entity recognition and argument association actions can be executed alternately in a interleaving order, in a psycholinguistically motivated left to right reading process.

Results on standard ACE2005 benchmarks show the advantages of solving event extraction as a single task. Our transition-based model gives the best-reported results in the literature. To our knowledge, we are the first to investigate transition-based methods for joint entity and event extraction.

2 Related Work

For pipelined event extraction, early studies [Grishman *et al.*, 2005; Ji and Grishman, 2008; Liao and Grishman, 2010; McClosky *et al.*, 2011] rely on manually designed indicator features. Recent work alleviate feature sparsity by using CNN [Chen *et al.*, 2015] and graph CNN [Nguyen and Grishman, 2018]. There are methods that jointly extract event mentions, which include structured predictions with global features [Li *et al.*, 2013; Li *et al.*, 2014; Judea and Strube, 2016], parameter sharing [Nguyen *et al.*, 2016; Sha *et al.*, 2018] and attention-based graph CNN [Liu *et al.*, 2018b]. However, the above work conduct argument identification by using external tools, which leads to error propagation.

Joint methods most related to ours include Yang and Mitchell [2016] and Nguyen and Nguyen [2019]. As mentioned in the introduction, both methods follow a trigger $\rightarrow$ entity $\rightarrow$ argument recognition decoding order. In contrast, we achieve fully joint decoding with interleaving actions for all the three subtasks, thereby achieving better information combination. Another limitation of Yang and Mitchell [2016] is that they heavily rely on feature templates struggling to capture sufficient discriminative information. As a result, their joint model can be suparssed by a pipelined but densely represented models [Sha *et al.*, 2018].

Our work follows a line of work doing joint modeling and decoding using transition-based methods [Zhang and Clark, 2011; Nivre, 2008], including segmentation and normalization [Qian *et al.*, 2015], syntactic chunking [Lyu *et al.*, 2016], relation extraction [Wang *et al.*, 2018a]. We fill a gap in the literature by investigating joint event extraction. In light of list-based arc-eager algorithm [Choi and McCallum, 2013] and high promising results of neural-based parsers [Dyer *et al.*, 2015; Wang *et al.*, 2018b], we propose a first neural transition-based framework for entity and event extraction.

3 Model

The input of our task is a sentence represented as a sequence of words $S = w_1, ..., w_n$, and the output includes:

Transitions	Change of State
SHIFT	$\frac{([\sigma\|i], \delta, j\|\lambda, e, \beta, T, E, R)}{([\sigma\|i\|\delta\|j], [], \psi, e, \beta, T, E, R)}$
DUAL-SHIFT	$\frac{([\sigma\|i], \delta, j\|\lambda, e, [\beta], T, E, R)}{([\sigma\|i\|\delta\|j], [], \psi, e, [j\|\beta], T, E, R)}$
NO-PASS	$\frac{([\sigma\|i], \delta, j\|\lambda, e, \beta, T, E, R)}{(\sigma, [i\|\delta], j\|\lambda, e, \beta, T, E, R)}$
LEFT-PASS$_l$	$\frac{([\sigma\|i], \delta, j\|\lambda, e, \beta, T, E, R)}{(\sigma, [i\|\delta], j\|\lambda, e, \beta, T, E, R \cup \{(i \overset{l}{\leftarrow} j)\})}$
RIGHT-PASS$_l$	$\frac{([\sigma\|i], \delta, j\|\lambda, e, \beta, T, E, R)}{(\sigma, [i\|\delta], j\|\lambda, e, \beta, T, E, R \cup \{(i \overset{l}{\rightarrow} j)\})}$
DELETE	$\frac{([\sigma\|i], \delta, \lambda, e, [j\|\beta], T, E, R)}{([\sigma\|i], \delta, \lambda, e, \beta, T, E, R)}$
TRIGGER-GEN$_l$	$\frac{([\sigma\|i], \delta, \lambda, e, [j\|\beta], T, E, R)}{([\sigma\|i], \delta, j\|\lambda, e, \beta, T \cup \{j\}, E, R)}$
ENTITY-GEN$_l$	$\frac{([\sigma\|i], \delta, \lambda, [j\|e], \beta, T, E, R)}{([\sigma\|i], \delta, j\|\lambda, [j\|e], \beta, T, E \cup \{j\}, R)}$
ENTITY-SHIFT	$\frac{([\sigma\|i], \delta, \lambda, e, [j\|\beta], T, E, R)}{([\sigma\|i], \delta, \lambda, [j\|e], \beta, T, E, R)}$
ENTITY-BACK	$\frac{([\sigma\|i], \delta, \lambda, [j\|e], [\beta], T, E, R)}{([\sigma\|i], \delta, \lambda, [], [e_{[1:]}\|\beta], T, E, R)}$

Table 1: Transition actions, [:] is a slice operation where index starts from 0, ψ denotes Null variable.

- A set of entity mentions E, which include references to entites. We consider the standard PER, ORG, GPE, LOC, FAC, VEH, WEA entity types plus ACE VALUE and TIME expressions [Yang and Mitchell, 2016].
- A set of event triggers T, namely the key words that most clearly express an event occurrence, such like *"sentenced"* in Figure 1. We follow Li *et al.*, [2013] and Sha *et al.*, [2018], assuming that each trigger consists only one word or token[2].
- A set of event arguments R on the entity mentions that are involved in an event. We collapse 8 time-related types into one as in Yang and Mitchell [2016], which results in total 29 role subtypes.

3.1 Challenges

Transition systems have been designed for building the output structures of semantic dependency parsing [Wang *et al.*, 2018b] and relation extraction [Wang *et al.*, 2018a], which are to some extent similar to the structure of event extraction. However, existing transition systems cannot be directly applied for our task, which poses bigger challenges because an event trigger can be associated with multiple entity mentions and a single entity mention can participate in several events. In addition, VALUE and TIME expressions, which as

[2]One can simply extent our model to multiple words scenario.

Transitions	Preconditions of transition actions
LEFT-*	$(\lambda \neq \psi) \wedge (\sigma \neq [\,]) \wedge (j \in T) \wedge (i \in E)$
RIGHT-*	$(\lambda \neq \psi) \wedge (\sigma \neq [\,]) \wedge (j \in E) \wedge (i \in T)$
SHIFT	$(\lambda \neq \psi) \wedge (\sigma = [\,])$
DUAL-SHIFT	$(\lambda \neq \psi) \wedge (\sigma = [\,]) \wedge (j \in T)$
DELETE	$(\exists j \in \beta) \wedge (e = [\,])$
TRIGGER-GEN	$(\lambda = \psi) \wedge (\exists j \in \beta) \wedge (e = [\,]) \wedge (j \notin T)$
ENTITY-SHIFT	$(\lambda = \psi) \wedge (\exists j \in \beta)$
ENTITY-GEN	$(\lambda = \psi) \wedge (e \neq [\,]) \wedge (j \notin E)$
ENTITY-BACK	$(\lambda = \psi) \wedge (e \neq [\,]) \wedge (j \in E)$

Table 2: Preconditions of transition actions

specific to event extraction, are more likely to overlaps with other entities. For example in Figure 1, the VALUE expression *"10 years in jail"* overlap with the FAC entity *"jail"*, which adds to the challenges to transition systems. Besides entity overlapping, there are also trigger-entity overlapping, in the case of *"Former President Bill Clinton..."*, where the word *"Former"* is a TIME expression as well as a trigger word, which triggers *End-Position* event. We handle these challenges by desinging a novel transition system. Code is realeased at https://github.com/zjcerwin/TransitionEvent

3.2 Transition System

For ease of illustration, we first define several symbols used in our transition system. We use an index i to represent the location of words w_i, of triggers t_i and entities e_i in the sentence. Let an element ε_i refer to either a trigger t_i or an entity e_i. Formally, a transition state is defined as $s = (\sigma, \delta, \lambda, e, \beta, T, E, R)$, where σ is a stack holding processed elements, δ is a queue holding elements temporarily popped out of σ, which will be pushed back in the future, e is a stack storing partial entity mentions, and β is a buffer holding unprocessed words. T and E are labeled trigger arcs and entity mention arcs, respectively. R is a set of argument role arcs. λ is a single variable holding a reference to an element ε_j one at a time. A is a stack used to store the action history. During state transition, arcs will only be generated between the variable $\lambda(\varepsilon_j)$ and the top element of $\sigma(\varepsilon_i)$.

Our transition actions are summarized in Table 1. The first five actions are used to generate argument roles. In particular, LEFT-PASS$_l$ add arc between $\lambda(t_j)$ and $\sigma(e_i)$, RIGHT-PASS$_l$ add arc between $\lambda(e_j)$ and $\sigma(t_i)$. If no semantic role can be assigned between $\lambda(\varepsilon_j)$ and $\sigma(\varepsilon_i)$, NO-PASS is performed. Note that ε_i can be either e_i or t_i. SHIFT and DUAL-SHIFT are performed when no elements are in σ. To handle situations where a word is a trigger and also the first word of an entity, DUAL-SHIFT additionally copies the trigger word in λ and pushes it onto β. Note that the *-PASS actions are forbidden when λ is Null. DELETE simply pops the top word w_i off β. TRIGGER-GEN moves w_i from β to λ adding event label l_t.

The last three actions are designed to recognize nested entities, among which ENTITY-SHIFT moves the top word w_i from β to e; ENTITY-GEN summarizes all elements in e to a vector representation, adding an entity label l_e, and moving the representation to λ; ENTITY-BACK pops all words off e and pushes all except the bottom word back to β. We found that this design of entity actions can handle arbitrary type of nested entities, while keeping minimum numbers of action steps necessary.

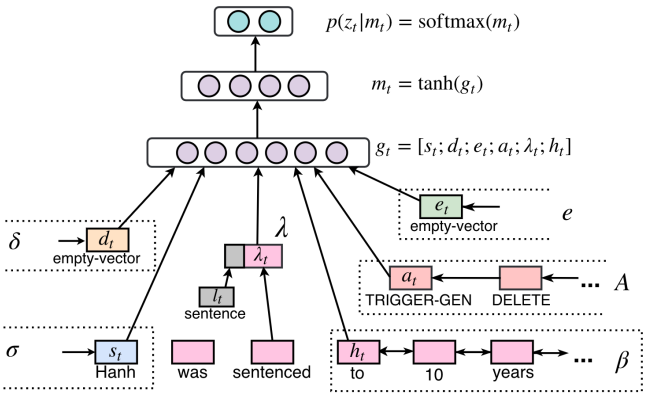

Figure 2: Action prediction model, where the transition state corresponds to state 7 in Table 3.

Given a certain transition state, only a subset of actions are legal, which can lead to a valid graph structure. We list the action preconditions in Table 2. To extract the three subtasks in proper order, we design the preconditions of all actions other than DELETE according to the state of λ. For example, if λ is not Null, only argument related actions are allowed. In addition, we also add type constraints between entities and triggers in a decoding state, such that a *Divorce* event can only happen with PER entities.

The gold-standard sequence of transitions for the input sentence in Figure 1 can be found in Table 3, where initial state is $([\,], [\,], \psi, [\,], [1, ..., 8],)$ and the terminal state is $(\sigma, \delta, \psi, [\,], [\,], T, E, R)$.

3.3 Method

We use neural network to learn dense representations of a transition state, for predicting the next action.

Input Representation

Formally, the representation for each word w_i, is a combination of four different types of vectors:

$$x_i = [v_i^{\text{w}}; v_i^{\text{pos}}; v_i^{\text{char}}; \text{BERT}_i]$$

where v_i^{w} denote a word embedding initalized with a pre-trained 100D Glove matrix[3], and v_i^{pos} denotes a randomly initalized POS tag embedding. For the i-th word, v_i^{char} denotes its character-level representation learned by using a Bi-LSTM [Lample *et al.*, 2016]. BERT$_i$ denotes a contextualized embedding by using the top layer of the uncased-base BERT model [Devlin *et al.*, 2018].

To capture semantic features of input sequences, we use two vanilla LSTM layers to encode x_i, which allows the model to capture long-term dependencies between words:

$$\overrightarrow{h_i} = \text{LSTM}_w(x_i, \overrightarrow{h}_{i-1})$$
$$\overleftarrow{h_i} = \text{LSTM}_w(x_i, \overleftarrow{h}_{i+1})$$

[3]https://nlp.stanford.edu/projects/glove/

State	Transition	σ	δ	λ	e	β	T	E	R
0	Initialization	[ ]	[ ]	Null	[ ]	[1,..., 8]	ϕ	ϕ	ϕ
1	ENTITY-SHIFT	[ ]	[ ]	Null	[1]	[2,..., 8]			
2	ENTITY-GEN	[ ]	[ ]	1^*	[1]	[2,..., 8]		$E \cup \{1 - \text{PER} \rightarrow 1\}$	
3	SHIFT	[1^*]	[ ]	Null	[1]	[2,..., 8]			
4	ENTITY-BACK	[1^*]	[ ]	Null	[ ]	[2,..., 8]			
5	DELETE	[1^*]	[ ]	Null	[ ]	[3,..., 8]			
6	TRIGGER-GEN	[1^*]	[ ]	$3^\diamond$	[ ]	[4,..., 8]	$T \cup \{3 - \text{Sentence} \rightarrow 3\}$		
7	LEFT-PASS	[ ]	[1^*]	$3^\diamond$	[ ]	[4,..., 8]			$R \cup \{3 - \text{Defendant} \rightarrow 1\}$
8	SHIFT	[1^*, $3^\diamond$]	[ ]	Null	[ ]	[4,..., 8]			
9	DELETE	[1^*, $3^\diamond$]	[ ]	Null	[ ]	[5,..., 8]			
10	ENTITY-SHIFT	[1^*, $3^\diamond$]	[ ]	Null	[5]	[6, 7, 8]			
11	ENTITY-SHIFT	[1^*, $3^\diamond$]	[ ]	Null	[5, 6]	[7, 8]			
12	ENTITY-SHIFT	[1^*, $3^\diamond$]	[ ]	Null	[5, 6, 7]	[8]			
13	ENTITY-SHIFT	[1^*, $3^\diamond$]	[ ]	Null	[5, 6, 7, 8]	[ ]			
14	ENTITY-GEN	[1^*, $3^\diamond$]	[ ]	5^*	[5, 6, 7, 8]	[ ]		$E \cup \{5 - \text{VAL} \rightarrow 8\}$	
15	RIGHT-PASS	[1^*]	[$3^\diamond$]	5^*	[5, 6, 7, 8]	[ ]			$R \cup \{5 - \text{Sentence} \leftarrow 3\}$
16	NO-PASS	[ ]	[$3^\diamond$, 1^*]	5^*	[5, 6, 7, 8]	[ ]			
17	SHIFT	[1^*, $3^\diamond$, 5^*]	[ ]	Null	[5, 6, 7, 8]	[ ]			
18	ENTITY-BACK	[1^*, $3^\diamond$, 5^*]	[ ]	Null	[ ]	[6, 7, 8]			
19	DELETE	[1^*, $3^\diamond$, 5^*]	[ ]	Null	[ ]	[7, 8]			
20	DELETE	[1^*, $3^\diamond$, 5^*]	[ ]	Null	[ ]	[8]			
21	ENTITY-SHIFT	[1^*, $3^\diamond$, 5^*]	[ ]	Null	[8]	[ ]			
22	ENTITY-GEN	[1^*, $3^\diamond$, 5^*]	[ ]	8^*	[8]	[ ]		$E \cup \{8 - \text{FAC} \rightarrow 8\}$	

Table 3: Transition sequence for the entity and event extraction in Figure 1, $\diamond$ indicates a trigger, $*$ indicates an entity, states 23-27 are omitted for brevity.

The forward and backward representations are concatenated to obtain a bi-directional representation $h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$.

State Representation

The buffer β is initalized by pushing all input words and their Bi-LSTM hidden vectors onto β in the reverse order. For representing the stacks σ, δ, e and A, we use StackLSTM [Dyer *et al.*, 2015]. By maintaining a stack pointer, StackLSTM allows *poping* elements off a sequence in a neural manner. Formally, the state of the stack σ at step t is computed as:

$$s_t = \text{StackLSTM}[\lambda_0, ..., \lambda_t]$$

where λ_i denotes the representation of currently recognized entity mention or trigger word, which is computed when ENTITY-GEN or TRIGGER-GEN is executed. We adopt two composition functions to recursively integrate label information of entities and triggers into the transition system as:

$$\lambda_i^{\text{entity}} = \tanh(W_{e,\lambda}[e_t; l_t^{\text{entity}}] + b_{e,\lambda})$$
$$\lambda_i^{\text{trigger}} = \tanh(W_{t,\lambda}[h_t; l_t^{\text{trigger}}] + b_{t,\lambda})$$

where $W_{e,\lambda}$ and $W_{t,\lambda}$ denote the learnable parameters for entity representation e_i and trigger representation h_i, respectively, while l_t^{entity} and l_t^{trigger} denote their type vectors. Note that this type of composition function is also applied for argument roles. We omit the equations for brevity.

Finally, as shown in Figure 2, we represent the model state at time t, including $(\sigma, \delta, e, A, \beta)$ and λ, as:

$$g_t = [s_t; d_t; e_t; a_t; \lambda_t; h_t]$$

All the states, including λ, have a private trainable vector representing no element being held for the current time step.

3.4 Action Prediction

To predict the current action at time step t, we first squeeze the state representation g_t to a lower-dimensional vector m_t by employing a fully-connected layer, and then use a softmax output layer to compute the action probability:

$$m_t = \tanh(W_m g_t + b_m) \quad (1)$$

$$p(z_t|m_t) = \frac{\exp(u_{z_t}^\top m_t + b_z)}{\sum_{z' \in \nu(S,A)} \exp(u_{z'}^\top m_t + b_{z'})} \quad (2)$$

where W_m denotes a learnable parameter matrix, u_z denotes a learnable column vector for transition action z, b_m and b_z are corresponding bias terms. The set $\nu(S, A)$ represents the set of valid candidate actions. For efficient decoding, we maximize the probability of the action sequence by greedily taking the action with highest score $p(z_t|m_t)$.

3.5 Training

We convert gold output structures in a set of training data into gold sequence of transition actions. For each transition state, we minimize the negative log-likehood of the corresponding gold action:

$$L(\theta) = -\frac{1}{T} \sum_t \log p(z_t|m_t; \theta) + \frac{\xi}{2} ||\theta_F||^2$$

where T is the size of the gold action sequence, θ is the parameter set of our network and ξ is a l_2 regularization term for output feed-forward network parameter set θ_F, which is drawn from Equations (1) and (2).

Model	Event Trigger Identification			Event Trigger Classification			Event Argument Identification			Argument Role Classification		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
StagedMaxEnt‡	73.9	66.5	70.0	70.4	63.3	66.7	**75.7**	20.2	31.9	**71.2**	19.0	30.0
TwoStageBeam‡	76.6	58.7	66.5	74.0	56.7	64.2	74.6	25.5	38.0	68.8	23.5	35.0
Reranking‡	**77.6**	65.4	71.0	**75.1**	63.3	68.7	73.7	38.5	50.6	70.6	36.9	48.4
ThreeStageDMCNN*	72.4	**78.6**	75.4	69.5	**75.5**	72.3	42.1	56.2	48.1	34.1	45.4	38.9
TwoStageTransition	77.6	73.2	75.3	74.7	71.5	73.1	55.4	51.1	53.1	51.3	47.5	49.4
TwoStageGRU‡	72.7	65.9	69.1	70.4	63.9	67.0	61.7	42.1	50.1	46.0	31.4	37.4
TwoStageNP‡	-	-	-	-	-	69.6	-	-	57.2	-	-	50.1
MultitaskGRU‡	70.5	74.5	72.5	68.0	71.8	69.8	59.9	**59.8**	**59.9**	52.1	**52.1**	52.1
JointTransition	76.7	75.5	**76.1**	74.4	73.2	**73.8**	60.0	55.1	57.4	55.7	51.1	**53.3**

Table 4: Event extraction results on the ACE2005 test set."‡" represents the systems which use dependency relation features.

4 Experiments

4.1 Settings

Dataset and Evaluation We perform experiments on the ACE2005 corpus. Following previous work [Li *et al.*, 2013; Yang and Mitchell, 2016; Sha *et al.*, 2018; Nguyen and Nguyen, 2019], we use 40 documents in the newswire category for the test set, 30 other documents in different categories for development, and 529 remaining documents for training. Stanford CoreNLP[4] is utilized to preprocess the data, including tokenization, POS-tagging and recognizing the "true" case of words. We adopt the F1-metric for evaluating the correctness of event extraction following [Li *et al.*, 2013; Yang and Mitchell, 2016; Nguyen and Nguyen, 2019], and paired t-test is used for measuring significance values.

Hyperparameters and Training Details We tune all the hyper-parameters on the standard development set. Dropout is adopted to mitigate overfitting, with a rate of 0.45 for word embeddings and 0.15 for hidden states. We use the Adam optimizer, employing a cosine learning rate decay strategy [Loshchilov and Hutter, 2016]. The maximum and the minimum learning rates are $1.4e^{-3}$ and $1e^{-4}$, respectively, and the restart increase factor is 2. The hidden and batch sizes are set to 140 and 32, respectively. The regularization term is set to $\xi = 1e^{-4}$. Finally, to combat unknown words during testing, we replace singleton words with a UNK embedding, with a probability of 0.5.

4.2 Results

Baselines

We compare our proposed method with several strong baselines in terms of micro F1-measure. The models are divided into statistical methods and neural methods. For the former, entity candidates are obtained by selecting the k-best ($k = 50$) output of a **CRFEntity** extractor [Yang and Mitchell, 2016]. These methods include: (1) **StagedMaxEnt** [Yang and Mitchell, 2016], a typical piplined method; (2) **TwoStageBeam** [Li *et al.*, 2013], a pipelined system with structure perception and global features; and (3) **Reranking** [Yang and Mitchell, 2016], the state-of-the-art statistical model mentioned in the introduction, which reranks the k-best outputs of two CRF taggers by performing joint inference within a document context.

For our transition-based model, we build two cascaded approaches as baselines: (4) **TwoStageTransition** is a two-stage transition model of this work, which first extracts entities (denote as **EntityTransition**) and then jointly detects triggers and argument roles. To train this model, we keep task related actions and states from Section 3.2, while the same input representations are used across the tasks; (5) **ThreeStageDMCNN**, our reimplementation of Chen *et al.*, [2015], which obtains entities by **EntityTransition**, with two separated CNN being conducted for event extraction.

Recently proposed neural models are also listed for comparison: (6) **TwoStageGRU** [Nguyen and Nguyen, 2019] is a two-stage baseline neural model, where entities are predicted using a sequence tagger, and a Bi-GRU with manually designed features is used for event extraction; (7) **TwoStageNP** [Sha *et al.*, 2018] combines dependency relations and Bi-LSTM for end-to-end learning. Their method build candidate arguments by choosing the NP phrases from an external parser. (8) **MultitaskGRU** [Nguyen and Nguyen, 2019] is a multitask model that performs all the subtasks by sharing Bi-GRU hidden representations. This method currently has the state-of-the-art performance on ACE2005 dataset. Binary features inherented from Li *et al.*, [2013] and Nguyen *et al.*, [2016] are also incorporated in this method.

The Advantage of Joint Modeling

Table 4 shows the results. With predicted entity mentions, our **JointTransition** shows consist performance improvements over the baselines **TwoStageTransition** and **ThreeStageDMCNN** on both trigger detection and argument detection. This indicates that modeling the dependencies of entities and events is effective for reducing error propagation. Compared to the state-the-of-art joint models, we can see that **JointTransition** significantly outperforms **Reranking** by 5.1% absolute F-score in triggers ($p < 0.03$) and 4.9% in arguments ($p < 0.03$), respectively, demonstrating the superiority of a one-stage model and neural representations. By allowing entity information propagate through transition states, **JointTransition** shows better performance than **MultitaskGRU**, boosting the precision of argument role by 3.6%. This demonstrates the advantage of joint learning and

[4]http://stanfordnlp.github.io/CoreNLP/

Model	P	R	F1
CRFEntity	85.5	73.5	79.1
Reranking	82.4	79.2	80.7
PipelineGRU	80.6	80.3	80.4
MultitaskGRU	82.0	80.4	81.2
EntityTransition	86.4	86.0	86.2
JointTransition	**88.2**	**88.1**	**88.1**

Table 5: Entity extraction results on the ACE2005 test set.

Model	P	R	F1
DMCNN [Chen *et al.*, 2015]	75.6	63.6	69.1
JRNN [Nguyen *et al.*, 2016]	66.0	73.0	69.3
dbRNN [Sha *et al.*, 2018]	74.1	69.8	71.9
HBTNGMA [Chen *et al.*, 2018]	**77.9**	69.1	73.3
JMEE [Liu *et al.*, 2018a]	76.3	71.3	73.7
JointTransition	74.4	**73.2**	**73.8**
JointTransition (*-BERT*)	73.8	68.8	71.2

Table 6: Event trigger classification result on the ACE2005 test set. The first group uses gold entities; the second uses predicted entites.

decoding as compared to joint learning and piplined decoding.

Regarding argument prediction, we find that the F-score for argument identification by **TwoStageNP** is 57.2%, which falls to 50.1% for argument role classification, with a 7.1% decrease. Similarly, **MultitaskGRU** sees a 7.8% decrease. This is saliently larger compared to a 2.2% decrease by the statistical method **Reranking**. One reason may be that the use of hand-crafted features by a statistical model effectively eliminates false positive roles, leading to a higher precision. In contrast, isolated decoding of **TwoStageNP** and **MultitaskGRU** can result in confusion between certain roles, such as *Place* vs *Destination* and *Origin* vs *Destination*, as observed by Nguyen and Nguyen [2019].

In contrast, our method gives much lower accuracy decrease compared with other neural methods, thanks to the use of joint decoding and composition functions. In particular, when each argument role decision is made, the existing partial graph is available as a source of features, which gives our model more informed role classification.

4.3 Analysis

Entity

In addition to extracting event mentions, our transition methods also extracts nested entity mentions. We compare its results with our pipeline method as well as previous work. First, we can see from Table 5 that **EntityTransition** largely outperforms **Reranking** and **MultitaskGRU**, wich boosts both precision and recall by an absolute 4% improvement ($p < 0.03$). The improvement comes from two factors: (1) our model captures rich input feature representations by introducing character-level Bi-LSTM and pre-trained language models; (2) By using specially designed actions, our transition system can handle overlapping entities. Few prior methods take this into account, which takes 5% of the total entities. Second, **JointTransition** further improves **EntityTransition** to 88.1% F-score, suggesting that jointly decoding is beneficial not only for events, but also for entity recognition.

Trigger Classification

We compare **JointTransition** with the state-of-the-art event trigger detection systems in Table 6. Even though gold entities are available by the baseline models, our model still shows competitive results, which indicates the ability to disambiguate context semantics despite noise in predictions of entites. We also show the results where BERT embeddings are removed from the input. It can be seen that the results are also comparable with existing event systems, indicating the improvement provided by the transition method itself is more noticeable.

Case Study

To further understand the effectiveness of our joint decoding model compared to previous joint model, we examine three cases, from which different aspects of models are reflected.

1. *... acquire all shares in GE Edison Life Insurance ...* **take** *over GE 's US car and* **fire** *insurance operations, the reports said.*

The word *"fire"* in sentence 1 is misclassifed to a *Attack* event by **MultitaskGRU**, because of its surrounding word *"car"*. In contrast, with the help of the *transfer-ownership* event from the word *"take"* and the role *Seller* from the entity *"GE"*, **JointTransition** is able to circumvent the noisy context.

2. *It is the first time they have had freedom of movement with* **cars** *and* **weapons** *since the start of the* **intifada** *...*

Some trigger words never appear in the training dataset, such as the word *"intifada"* in case 2. As a result, it is difficult for the methods **Reranking** and **MultitaskGRU** to associate it with *Attack* event. However, thanks to pre-trained language models and joint decoding, our method correctly infers the event trigger.

3. *According to one report, he* **received** *3.5 million dollars for the* **film rights** *to his latest book.*

In case 3, our model misidentifies that the word *"received"* triggers a *transfer-money* event, owing to failure in detecting the fact that the key phrase of *"film rights"* belongs to the stock field. This results in false detections of two argument roles. This type of incorrect predictions, which accounts for 38.8% of our error cases, can be potentially alleviated by using explicit dependency features, which can inform the model that *"received"* is directly connected to *"film rights"* in the corresponding parse tree.

5 Conclusion

We introduced a novel transition-based model for jointly predicting nested entities, event triggers, as well as their semantic roles in event extraction. Unlike previous methods for event extraction, which detect entities and event mentions in multiple stages or separated tasks, our method captures structural dependencies among entities and event mentions by using a incremental left to right reading order. Experimental results on the ACE2005 benchmark show that our model achieves the state-of-the-art performance.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and suggestions. We also would like to thank Yu Hong for providing the ACE2005 corpus. Work was done when the first author was visiting Westlake University. Mengchi Liu and Donghong Ji are the corresponding authors.

References

[Chen *et al.*, 2015] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*, 2015.

[Chen *et al.*, 2018] Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *EMNLP*, 2018.

[Choi and McCallum, 2013] Jinho D Choi and Andrew McCallum. Transition-based dependency parsing with selectional branching. In *ACL*, 2013.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Ding *et al.*, 2014] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Using structured events to predict stock price movement: An empirical investigation. In *EMNLP*, 2014.

[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 *ACL*, 2015.

[Grishman *et al.*, 2005] Ralph Grishman, David Westbrook, and Adam Meyers. Nyu's english ace 2005 system description. *ACE*, 2005.

[Hong *et al.*, 2011] Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. Using cross-entity inference to improve event extraction. In *ACL*, 2011.

[Ji and Grishman, 2008] Heng Ji and Ralph Grishman. Refining event extraction through cross-document inference. *ACL-08*, 2008.

[Judea and Strube, 2016] Alex Judea and Michael Strube. Incremental global event extraction. In *COLING*, 2016.

[Lample *et al.*, 2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *NAACL-HLT*, 2016.

[Li *et al.*, 2013] Qi Li, Heng Ji, and Liang Huang. Joint event extraction via structured prediction with global features. In *ACL*, 2013.

[Li *et al.*, 2014] Qi Li, Heng Ji, HONG Yu, and Sujian Li. Constructing information networks using one single model. In *EMNLP*, 2014.

[Liao and Grishman, 2010] Shasha Liao and Ralph Grishman. Using document level cross-event inference to improve event extraction. In *ACL*, 2010.

[Liu *et al.*, 2018a] Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. Event detection via gated multilingual attention mechanism. In *AAAI*, 2018.

[Liu *et al.*, 2018b] Xiao Liu, Zhunchen Luo, and Heyan Huang. Jointly multiple events extraction via attention-based graph information aggregation. In *EMNLP*, 2018.

[Loshchilov and Hutter, 2016] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[Lyu *et al.*, 2016] Chen Lyu, Yue Zhang, and Donghong Ji. Joint word segmentation, pos-tagging and syntactic chunking. In *AAAI*, 2016.

[McClosky *et al.*, 2011] David McClosky, Mihai Surdeanu, and Christopher D Manning. Event extraction as dependency parsing. In *ACL*, 2011.

[Nguyen and Grishman, 2018] Thien Huu Nguyen and Ralph Grishman. Graph convolutional networks with argument-aware pooling for event detection. In *AAAI*, 2018.

[Nguyen and Nguyen, 2019] Trung Minh Nguyen and Thien Huu Nguyen. One for all: Neural joint modeling of entities and events. In *AAAI*, 2019.

[Nguyen *et al.*, 2016] Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. Joint event extraction via recurrent neural networks. In *NAACL*, 2016.

[Nivre, 2008] Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 2008.

[Qian *et al.*, 2015] Tao Qian, Yue Zhang, Meishan Zhang, Yafeng Ren, and Donghong Ji. A transition-based model for joint segmentation, pos-tagging and normalization. In *EMNLP*, 2015.

[Sha *et al.*, 2018] Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *AAAI*, 2018.

[Srihari and Li, 2000] Rohini Srihari and Wei Li. A question answering system supported by information extraction. In *ANLP*, 2000.

[Wang *et al.*, 2018a] Shaolei Wang, Yue Zhang, Wanxiang Che, and Ting Liu. Joint extraction of entities and relations based on a novel graph scheme. In *IJCAI*, 2018.

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

[Yang and Mitchell, 2016] Bishan Yang and Tom Mitchell. Joint extraction of events and entities within a document context. In *NAACL-HLT*, 2016.

[Zhang and Clark, 2011] Yue Zhang and Stephen Clark. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 2011.