RESEARCH PAPER

Joint Entity and Event Extraction with Generative Adversarial Imitation Learning

Tongtao Zhang[1], **Heng Ji**[1†] **& Avirup Sil**[2]

[1]Computer Science Department, Rensselaer Polytechnic Institute, Troy, New York 12180-3590, USA

[2]IBM Research AI, Armonk, New York 10504-1722, USA

Keywords: Information extraction; Event extraction; Imitation learning; Generative adversarial network

Citation: T. Zhang, H. Ji, & A. Sil. Joint entity and event extraction with generative adversarial imitation learning. Data Intelligence 1(2019), 99-120. doi: 10.1162/dint_a_00014

Received: December 24, 2018; Revised: February 11, 2019; Accepted: February 19, 2019

ABSTRACT

We propose a new framework for entity and event extraction based on generative adversarial imitation learning—an inverse reinforcement learning method using a generative adversarial network (GAN). We assume that instances and labels yield to various extents of difficulty and the gains and penalties (rewards) are expected to be diverse. We utilize discriminators to estimate proper rewards according to the difference between the labels committed by the ground-truth (expert) and the extractor (agent). Our experiments demonstrate that the proposed framework outperforms state-of-the-art methods.

1. INTRODUCTION

Event extraction (EE) is a crucial information extraction (IE) task that focuses on extracting structured information (i.e., a structure of event trigger and arguments, "*what is happening*", and "*who or what is involved*") from unstructured texts. For example, in the sentence "*Masih's alleged comments of blasphemy are punishable by* ***death*** *under Pakistan Penal Code*" shown in Figure 1, there is a Sentence event ("punishable"), and an Execute event ("*death*") involving the person entity "*Masih*". Most event extraction research has been in the context of the 2005 National Institute of Standards and Technology (NIST) Automatic Content Extraction (ACE) sentence-level event mention task [1], which also provides the standard corpus. The annotation guidelines of the ACE program define an event as a specific occurrence of something that happens involving participants, often described as a change of state [2]. More recently, the NIST Text

† Corresponding author: Heng Ji (Email: jih@rpi.edu; ORCID: 0000-0002-7954-7994).

Analysis Conference's Knowledge Base Population (TAC-KBP) community has introduced document-level event argument extraction shared tasks for 2014 and 2015 (KBP EA).

In the last five years, many event extraction approaches have brought forth encouraging results by retrieving additional related text documents [3], introducing rich features of multiple categories [4, 5], incorporating relevant information within or beyond context [6, 7, 8, 9] and adopting neural network frameworks [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20].

However, there are still challenging cases: for example, in the following sentences: "*Masih's alleged comments of blasphemy are punishable by* ***death*** *under Pakistan Penal Code*" and "*Scott is charged with first-degree homicide for the* ***death*** *of an infant*", the word **death** can trigger an Execute event in the former sentence and a Die event in the latter one. With similar local information (word embeddings) or contextual features (both sentences include legal events), supervised models pursue the probability distribution which resembles that in the training set (e.g., we have overwhelmingly more Die annotation on death than Execute), and will label both as a Die event, causing an error in the former instance.

Such a mistake is due to the lack of a mechanism that explicitly deals with wrong and confusing labels. Many multi-classification approaches utilize cross-entropy loss, which aims at boosting the probability of the correct labels, and usually treats wrong labels equally and merely inhibits them indirectly. Models are trained to capture features and weights to pursue correct labels, but will become vulnerable and unable to avoid mistakes when facing ambiguous instances, where the probabilities of the confusing and wrong labels are not sufficiently "suppressed". Therefore, exploring information from wrong labels is a key to make the models robust.

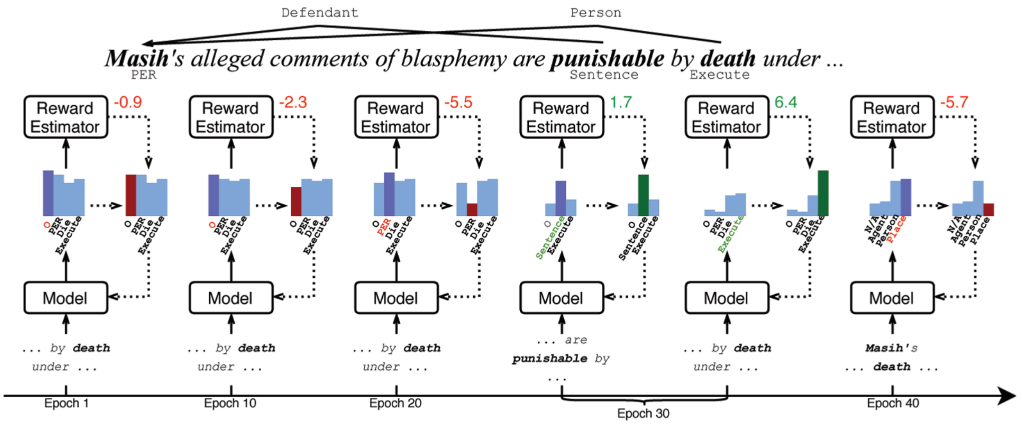


Figure 1. Our framework includes a reward estimator based on a generative adversarial network (GAN) to issue dynamic rewards with regard to the labels (actions) committed by event extractor (agent). The reward estimator is trained upon the difference between the labels from ground truth (expert) and extractor (agent). If the extractor repeatedly misses Execute label for "*death*", the penalty (negative reward values) is strengthened; if the extractor makes surprising mistakes: label "*death*" as Person or label Person "*Masih*" as Place role in Sentence event, the penalty is also strong. For cases where extractor is correct, simpler cases such as Sentence on "*death*" will take a smaller gain while difficult cases Execute on "*death*" will be awarded with larger reward values.

In this paper, to combat the problems of previous approaches toward this task, we propose a dynamic mechanism—inverse reinforcement learning—to directly assess correct and wrong labels on instances in entity and event extraction. We assign explicit scores on cases—or **rewards** in terms of Reinforcement Learning (RL). We adopt discriminators from a generative adversarial network (GAN) to estimate the reward values. Discriminators ensure the highest reward for ground-truth (expert) and the extractor attempts to imitate the expert by pursuing highest rewards. For challenging cases, if the extractor continues selecting wrong labels, the GAN keeps expanding the margins between rewards for ground-truth labels and (wrong) extractor labels and eventually deviates the extractor from wrong labels.

The main contributions of this paper can be summarized as follows:

- We apply RL framework to event extraction tasks, and the proposed framework is an end-to-end and pipelined approach that extracts entities and event triggers and determines the argument roles for detected entities.
- With inverse RL propelled by the GAN, we demonstrate that a dynamic reward function ensures more optimal performance in a complicated RL task.

2. RELATED WORK

One of the recent event extraction approaches mentioned in the introductory section [18] utilizes the GAN in event extraction. The GAN in the cited work outputs generated features to regulate the event model from features leading to errors, while our approach directly assesses the mistakes to explore levels of difficulty in labels. Moreover, our approach also covers argument role labeling, while the cited paper does not.

RL-based methods have been recently applied to a few information extraction tasks such as relation extraction, and both relation frameworks from [21, 22] apply RL on entity relation detection with a series of predefined rewards.

We are aware that the term *imitation learning* is slightly different from *inverse reinforcement learning*. Techniques of imitation learning [23, 24, 25] attempt to map the states to expert actions by following demonstration, which resembles supervised learning, while inverse RL [26, 27, 28, 29, 30] estimates the rewards first and applies the rewards to RL. [31] is an imitation learning application on bio-medical event extraction, and there is no reward estimator used. We humbly recognize our work as inverse reinforcement learning approach although "Generative Adversarial Imitation Learning" (GAIL) is named after imitation learning.

3. TASK AND TERM PRELIMINARIES

In this paper we follow the schema of Automatic Content Extraction (ACE) [1] to detect the following elements from unstructured natural language data:

- Entity: Word or phrase that describes a real world object such as a person ("*Masih*" as PER in Figure 1). ACE schema defines seven types of entities.
- Event trigger: The word that most clearly expresses an event (interaction or change of status). ACE schema defines 33 types of events such as Sentence ("*punishable*" in Figure 1) and Execute ("*death*").
- Event argument: An entity that serves as a participant or attribute with a specific role in an event mention, in Figure 1 e.g., a PER "*Masih*" serves as a Defendant in a Sentence event triggered by "*punishable*".

The ACE schema also comes with a data set—ACE2005[①]—which has been used as a benchmark for information extraction frameworks and we will introduce this data set in Section 6.

For broader readers who might not be familiar with RL, we briefly introduce their counterparts or equivalent concepts in supervised models with the RL terms in the parentheses: our goal is to train an extractor (agent A) to label entities, event triggers and argument roles (actions a) in text (environment e); to commit correct labels, the extractor consumes features (state s) and follows the ground truth (expert E); a reward R will be issued to the extractor according to whether it is different from the ground truth and how serious the difference is—as shown in Figure 1, a repeated mistake is definitely more serious—and the extractor improves the extraction model (policy π) by pursuing maximized rewards.

Our framework can be briefly described as follows: given a sentence, our extractor scans the sentence and determines the boundaries and types of entities and event triggers using Q-Learning (Section 4.1); meanwhile, the extractor determines the relations between triggers and entities—*argument roles* with policy gradient (Section 4.2). During the training epochs, GANs estimate rewards which stimulate the extractor to pursue the most optimal joint model (Section 5).

4. FRAMEWORK AND APPROACH

4.1 Q-Learning for Entities and Triggers

The entity and trigger detection is often modeled as a sequence labeling problem, where long-term dependency is a core nature; and RL is a well-suited method [32].

From RL perspective, our extractor (agent A) is exploring the *environment*, or unstructured natural language sentences when going through the sequences and committing labels (actions a) for the tokens. When the extractor arrives at the tth token in the sentence, it observes information from the environment and its previous action a_{t-1} as its current *state* s_t; when the extractor commits a current action a_t and moves to the next token, it has a new state s_{t+1}. The information from the environment is the token's context embedding v_t, which is usually acquired from Bidirectional Long Short-Term Memory (Bi-LSTM) [33]

[①] https://catalog.ldc.upenn.edu/LDC2006T06 outputs; previous action a_{t-1} may impose some constraint for current action a_t, e.g., I–ORG does not follow B–PER[②]. With the aforementioned notations, we have

$$s_t = < v_t, a_{t-1} > . \tag{1}$$

To determine the current action a_t, we generate a series of *Q-Tables* with

$$Q_{sl}(s_t, a_t) = \boldsymbol{f}_{sl}(s_t | s_{t-1}, s_{t-2}, \ldots, a_{t-1}, a_{t-2}, \ldots), \tag{2}$$

where $\boldsymbol{f}_{sl}(\cdot)$ denotes a function that determines the **Q-values** using the current state as well as previous states and actions. Then we achieve

$$\hat{a}_t = \arg\max_{a_t} Q_{sl}(s_t, a_t). \tag{3}$$

Equations (2) and (3) suggest that a Recurrent Neural Network (RNN)-based framework which consumes current input and previous inputs and outputs can be adopted, and we use a unidirectional LSTM as [34]. We have a full pipeline as illustrated in Figure 2.

For each label (action a_t) with regard to s_t, a reward $r_t = r(s_t, a_t)$ is assigned to the extractor (agent). We use Q-Learning to pursue the most optimal sequence labeling model (policy π) by maximizing the expected value of the sum of future rewards $\mathbb{E}(R_t)$, where R_t represents the sum of discounted future rewards $r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$ with a discount factor γ, which determines the influence between current and next states.

We utilize *Bellman Equation* to update the Q-value with regard to the current assigned label to approximate an optimal model (policy π^*):

$$Q_{sl}^{\pi^*}(s_t, a_t) = r_t + \gamma \max_{a_{t+1}} Q_{sl}(s_{t+1}, a_{t+1}). \tag{4}$$

As illustrated in Figure 3, when the extractor assigns a wrong label on the "*death*" token because the Q-value of Die ranks first, Equation (4) will penalize the Q-value with regard to the wrong label; while in later epochs, if the extractor commits a correct label of Execute, the Q-value will be boosted and make the decision reinforced.

We minimize the loss in terms of mean squared error between the original and updated Q-values notated as $Q'_{sl}(s_t, a_t)$:

$$L_{sl} = \frac{1}{n} \sum_{t}^{n} \sum_{a} (Q'_{sl}(s_t, a_t) - Q_{sl}(s_t, a_t))^2 \tag{5}$$

and apply back propagation to optimize the parameters in the neural network.

[②] In this work, we use BIO, e.g., "B–Meet" indicates the token is beginning of Meet trigger, "I–ORG" means that the token is inside an organization phrase, and "O" denotes null.

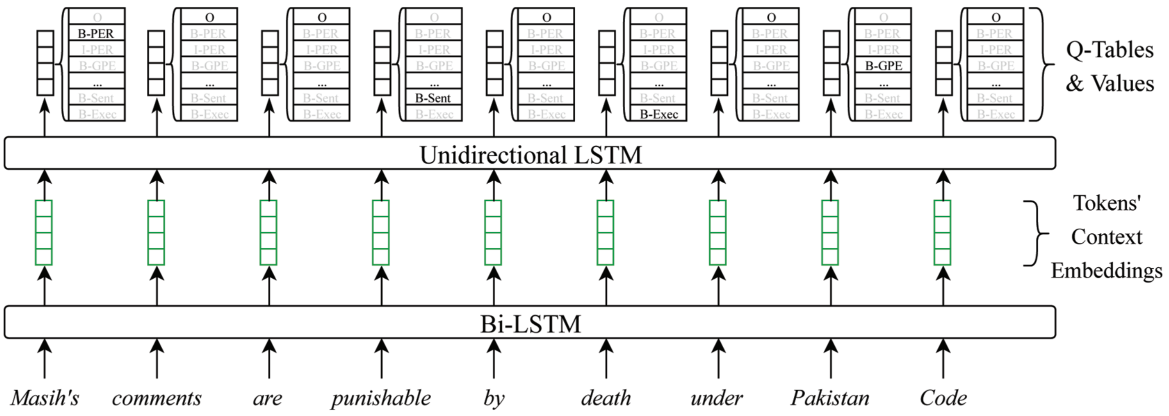
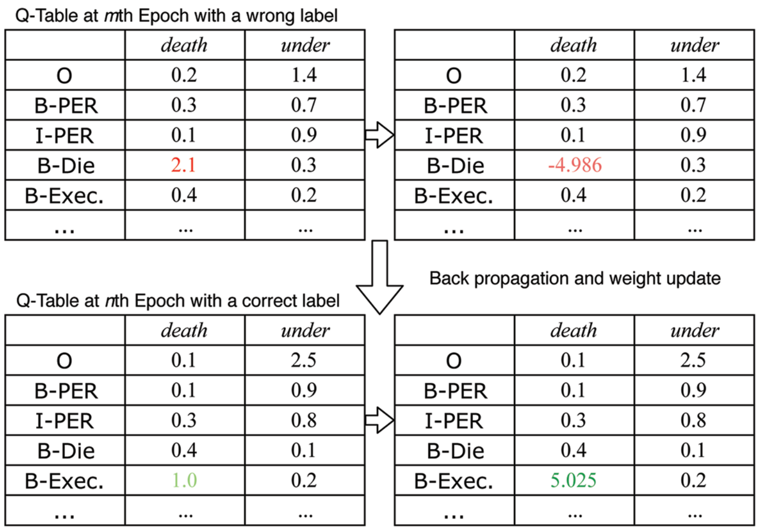


Figure 2. A pipeline from input sentence to sequence labels mentioned in Section 4.1. Q-Table and values for each current step is calculated using the unidirectional LSTM based on context embeddings of current and previous tokens as well as Q-Tables and values from previous steps. Context embeddings are calculated using Bi-LSTM from local token embeddings. Pre-trained embeddings based on Bi-LSTM such as ELMo [35] are also good candidates for context embeddings.


Q-Table at *m*th Epoch with a wrong label
Back propagation and weight update
Q-Table at *n*th Epoch with a correct label

	death	*under*
O	0.2	1.4
B-PER	0.3	0.7
I-PER	0.1	0.9
B-Die	2.1	0.3
B-Exec.	0.4	0.2
...	...	...

	death	*under*
O	0.2	1.4
B-PER	0.3	0.7
I-PER	0.1	0.9
B-Die	-4.986	0.3
B-Exec.	0.4	0.2
...	...	...

	death	*under*
O	0.1	2.5
B-PER	0.1	0.9
I-PER	0.3	0.8
B-Die	0.4	0.1
B-Exec.	1.0	0.2
...	...	...

	death	*under*
O	0.1	2.5
B-PER	0.1	0.9
I-PER	0.3	0.8
B-Die	0.4	0.1
B-Exec.	5.025	0.2
...	...	...

Figure 3. An illustrative example of updating the Q-values with Equation (4), with fixed rewards $r = \pm 5$ for correct/wrong labels and discount factor $\lambda = 0.01$. Score for a wrong label is penalized while correct one is reinforced.

4.2 Policy Gradient for Argument Roles

After the extractor determines the entities and triggers, it takes pairs of one trigger and one entity (argument candidate) to determine whether the latter serves a role in the event triggered by the former.

In this task, for each pair of trigger and argument candidate, our extractor observes the context embeddings of trigger and argument candidate—$v_{t_{tr}}$ and $v_{t_{ar}}$, respectively, as well as the output of another Bi-LSTM consuming the sequence of context embeddings between trigger and argument candidates in the state; the state also includes a representation (one-hot vector) of the entity type of the argument candidate $a_{t_{ar}}$, and the event type of the trigger $a_{t_{ar}}$ also determines the available argument role labels, e.g., an Attack event never has Adjudicator arguments as Sentence events. With these notations we have:

$$s_{tr,ar} = < v_{t_{tr}}, v_{t_{ar}}, a_{t_{tr}}, a_{t_{ar}}, \boldsymbol{f}_{ss} >, \tag{6}$$

where the footnote *tr* denotes the trigger, *ar* denotes argument candidate, and $\boldsymbol{f}_{ss}$ denotes the sub-sentence Bi-LSTM for the context embeddings between trigger and argument.

We have another ranking table for argument roles:

$$Q_{tr,ar}(s_{tr,ar}, a_{tr,ar}) = \boldsymbol{f}_{tr,ar}(s_{tr,ar}), \tag{7}$$

where $\boldsymbol{f}_{tr,ar}$ represents a mapping function whose output sizes are determined by the trigger event type $a_{t_{tr}}$, e.g., Attack event has five labels—Attacker, Target, Instrument, Place and Not-a-role and the mapping function for Attack event contains a fully-connected layer with output size of five.

And we determine the role with Equation (8):

$$\hat{a}_{tr,ar} = \underset{a_{tr,ar}}{\arg\max}\, Q_{tr,ar}(s_{tr,ar}, a_{tr,ar}). \tag{8}$$

We assign a reward $r(s_{tr,ar}, a_{tr,ar})$ to the extractor, and since there is one step in determining the argument role label, the expected values of $R = r(s_{tr,ar}, a_{tr,ar})$.

We utilize another RL algorithm—Policy Gradient [36] to pursue the most optimal argument role labeling performance.

We have probability distribution of argument role labels that are from the softmax output of Q-values:

$$P(a_{tr,ar}|s_{tr,ar}) = \mathrm{softmax}(Q_{tr,ar}(s_{tr,ar}, a_{tr,ar})). \tag{9}$$

To update the parameters, we minimize loss function with Equation (10):

$$L_{pg} = -R \log P(a_{tr,ar}|s_{tr,ar}). \tag{10}$$

From Equation (10) and Figure 4 we acknowledge that, when the extractor commits a correct label (Agent for the GPE entity "*Pakistan*"), the reward encourages $P(a_{tr,ar}|s_{tr,ar})$ to increase; and when the extractor is wrong (e.g., Place for "*Pakistan*"), the reward will be negative, leading to a decreased $P(a_{tr,ar}|s_{tr,ar})$.

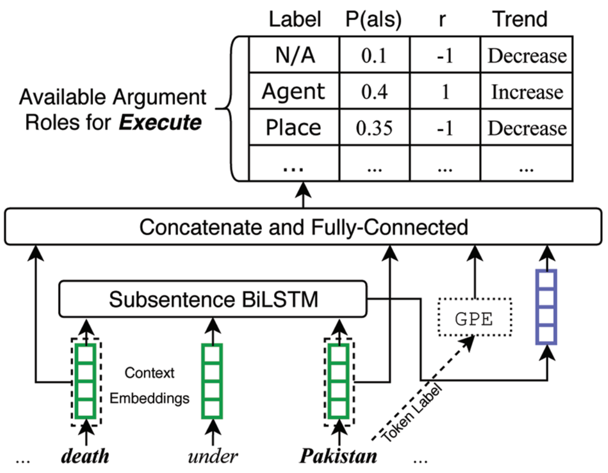


Figure 4. The extractor combines context embeddings of the trigger and entity, as well as a one-hot vector that represents entity type and Bi-LSTM output of sub-sentence between the trigger and argument. The column "trend" denotes the changes of $P(a_{tr,ar}|s_{tr,ar})$ after policy gradient optimization in Equation (10).

4.3 Choice of Algorithms

Here we have a brief clarification on different choices of RL algorithms in the two tasks.

In the sequence labeling task, we do not take policy gradient approach due to high variance of $\mathbb{E}(R_t)$, i.e., the sum of future rewards R_t should be negative when the extractor chooses a wrong label, but an ill-set reward and discount factor γ assignment or estimation may give a positive R_t (often with a small value) and still push up the probability of the wrong action, which is not desired. There are some variance reduction approaches to constraining the R_t but they still need additional estimation and bad estimation will introduce new risks. Q-Learning only requires rewards on current actions r_t, which are relatively easy to constrain.

In the argument role labeling task, determination on each trigger-entity pair consists of only one single step and R_t is exactly the current reward r, and then policy gradient approach performs correctly if we ensure negative rewards for wrong actions and positive for correct ones. However, this one-step property impacts the Q-Learning approach: without new positive values from further steps, a small positive reward on current correct labels may make the updated Q-value smaller than those wrong ones.

5. GENERATIVE ADVERSARIAL IMITATION LEARNING

So far in our paper, the reward values demonstrated in the examples are fixed, we have

$$r = \begin{cases} c_1 & \text{when } a \text{ is correct,} \\ c_2 & \text{otherwise,} \end{cases} \tag{11}$$

and typically we have $c_1 > c_2$.

This strategy makes the RL-based approach no difference from classification approaches with cross-entropy in terms of "treating wrong labels equally" as discussed in the introductory section. Moreover, recent RL approaches on relation extraction [21, 22] adopt a fixed setting of reward values with regard to different phases of entity and relation detection based on empirical tuning, which requires additional tuning work when switching to another data set or schema.

In event extraction task, entity, event and argument role labels yield to a complex structure with variant difficulties. Errors should be evaluated case by case, and from epoch to epoch. In the earlier epochs, when parameters in the neural networks are slightly optimized, all errors are tolerable, e.g., in sequence labeling, extractor within the first two or three iterations usually labels most tokens with O labels. As the epoch number increases, the extractor is expected to output more correct labels; however, if the extractor makes repeated mistakes—e.g., the extractor persistently labels "*death*" as O in the example sentence "*... are punishable by* ***death*** ..." during multiple epochs—or is stuck in difficult cases—e.g., whether FAC (facility) token "*bridges*" serves as a Place or Target role in an Attack event triggered by "*bombed*" in sentence "*US aircraft* ***bombed*** *Iraqi tanks holding* ***bridges***..."—a mechanism is required to assess these challenges and to correct them with salient and dynamic rewards.

We describe the training approach as a process of extractor (agent A) imitating the ground-truth (expert E), and during the process, a mechanism ensures that the highest reward values are issued to correct labels (actions a), including the ones from both expert E and a:

$$\mathbb{E}_{\pi E}\left[R(s,a)\right] \geq \mathbb{E}_{\pi A}\left[R(s,a)\right]. \tag{12}$$

This mechanism is Inverse Reinforcement Learning [26], which estimates the reward first in an RL framework.

Equation (12) reveals a scenario of adversary between ground truth and extractor and GAIL [29], which is based on GAN [37], fits such adversarial nature.

In the original GAN, a generator generates (fake) data and attempts to confuse a discriminator D which is trained to distinguish fake data from real data. In our proposed GAIL framework, the extractor (agent A) substitutes the generator and commits labels to the discriminator D; the discriminator D, now serves as reward estimator, aims to issue largest rewards to labels (actions) from the ground-truth (expert E) or identical ones from the extractor but provides lower rewards for other/wrong labels.

Rewards $R(s,a)$ and the output of D are now equivalent and we ensure:

$$\mathbb{E}_{\pi_E}\left[D\left(s,a_E\right)\right] \geq \mathbb{E}_{\pi A}\left[D\left(s,a_A\right)\right], \tag{13}$$

where s, a_E and a_A are inputs of the discriminator. In the sequence labeling task, s consists of the context embedding of current token v_t and a one-hot vector that represents the previous action a_{t-1} according to Equation (1), and in the argument role labeling task, s comes from the representations of all elements mentioned in Equation (6); a_E is a one-hot vector of ground-truth label (expert, or "real data") while a_A denotes the counterpart from the extractor (agent, or "generator"). The concatenated s and a_E is the input for "real data" channel while s and a_A build the input for "generator" channel of the discriminator.

In our framework, due to the different dimensions in the two tasks and event types, we have 34 discriminators (one for sequence labeling, and 33 for event argument role labeling with regard to 33 event types). Every discriminator consists of two fully-connected layers with a sigmoid output. The original output of D denotes a probability which is bounded in [0,1], and we use linear transformation to shift and expand it:

$$R\left(s,a\right) = \alpha * \left(D\left(s,a\right) - \beta\right), \tag{14}$$

e.g., in our experiments, we set $\alpha = 20$ and $\beta = 0.5$ and make $R(s,a) \in [-10,10]$.

To pursue Equation (13), we minimize the loss function and optimize the parameters in the neural network:

$$L_D = -\left(\mathbb{E}\left[\log D\left(s,a_E\right)\right] + \mathbb{E}\left[\log\left(1 - D\left(s,a_A\right)\right)\right]\right). \tag{15}$$

During the training process, after we feed the neural network mentioned in Section 4.1 and 4.2 with a mini-batch of the data, we collect the features (or states s), corresponding extractor labels (agent actions a_A) and ground-truth (expert actions a_E) to update the discriminators according to Equation (15); then we feed features and extractor labels into the discriminators to acquire reward values and train the extractor—or the generator from the GAN's perspective.

Since the discriminators are continuously optimized, if the extractor (generator) makes repeated mistakes or makes surprising ones (e.g., considering a PER as a Place), the margin of rewards between correct and wrong labels expands and outputs reward with larger absolute values. Hence, in sequence labeling task, the updated Q-values are updated with a more discriminative difference, and, similarly, in argument role labeling task, the $P(a|s)$ also increases or decreases more significantly with a larger absolute reward value.

Figure 5 illustrates how we utilize a GAN for reward estimation.

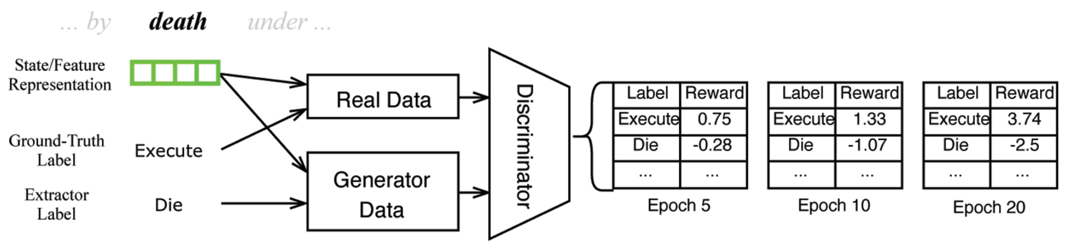


Figure 5. An illustrative example of the GAN structure in sequence labeling scenario (argument role labeling scenario has the identical frameworks except vector dimensions). As introduced in Section 5, the "real data" in the original GAN is replaced by feature/state representation (Equation (1), or Equation (6) for argument role labeling scenario) and ground-truth labels (expert actions) in our framework, while the "generator data" consists of features and extractor's attempt labels (agent actions). The discriminator serves as the reward estimator and a linear transform is utilized to extend the *D*'s original output of probability range [0,1].

In case where discriminators are not sufficiently optimized (e.g., in early epochs) and may output undesired values—e.g., negative for correct actions, we impose a hard margin:

$$\tilde{R}(s,a) = \begin{cases} \max\left(0.1, R(s,a)\right) & \text{when } a \text{ is correct,} \\ \min\left(-0.1, R(s,a)\right) & \text{otherwise} \end{cases} \tag{16}$$

to ensure that correct actions will always take positive reward values and wrong ones take negative.

6. EXPLORATION

In training phase, the extractor selects labels according to the rankings of Q-values in Equations (3) and (8) and GANs will issue rewards to update the Q-Tables and policy probabilities; and we also adopt ε-greedy strategy: we set a probability threshold $\varepsilon \in [0,1)$ and uniformly sample a number $\rho \in [0,1]$ before the extractor commits a label for an instance:

$$\hat{a} = \begin{cases} \arg\max_a Q(s,a), \text{ if } \rho \geq \varepsilon \\ \text{Randomly pick up an action, if others} \end{cases}$$

With this strategy, the extractor is able to explore all possible labels (including correct and wrong ones), and acquires rewards with regard to all labels to update the neural networks with richer information.

Moreover, after one step of ε-greedy exploration, we also force the extractor to commit ground-truth labels and issue it with expert (highest) rewards, and update the parameters accordingly. This additional step is inspired by [38, 39], which combines cross-entropy loss from supervised models with RL loss functions[③]. Such combination can simultaneously and explicitly encourage correct labels and penalize wrong labels and greatly improve the efficiency of pursuing optimal models.

③ We do not directly adopt this because we treat cross-entropy loss as fixed rewards with $r = 1$ for correct label and $r = 0$ for wrong label but we prioritize the dynamic rewards.

7. EXPERIMENTS

7.1 Experiment Setup

To evaluate the performance with our proposed approach, we utilize ACE2005 documents. To align with state-of-the-art frameworks such as [13, 16], we exclude informal documents from Conversational Telephone Speech (cts) and UseNet (un), and the rest of the documents include newswire (nw), weblogs (wl), broadcast news (bn) and broadcast conversations (bc) crawled between 2003 and 2005 and fully annonotated with 5,272 triggers and 9,612 arguments. To ensure fair comparison with the state-of-the-art methods, we follow the splits of training (529 documents with 14,180 sentences), validation (30 documents with 863 sentences) and test (40 documents with 672 sentences) data and adopt the same criteria of the evaluation:

- An entity (named entities and nominals) is correct if its entity type and offsets find a match in the ground truth.
- A trigger is correct if its event type and offsets find a match in the ground truth.
- An argument is correctly labeled if its event type, offsets and role find a match in the ground truth.
- All the aforementioned elements are evaluated using precision (denoted as P in the tables, the ratio of correct instances in the system result), recall (denoted as R in the tables, the ratio of correct system results in the ground-truth annotation) and F1 scores (denoted as F1, harmonic average of the precision and recall).

We use ELMo embeddings[④] [35]. Because ELMo is delivered with built-in Bi-LSTMs, we treat ELMo embedding as context embeddings in Figures 2 and 4. We use GAIL-ELMo in the tables to denote the setting.

Moreover, in order to disentangle the contribution from ELMo embeddings, we also present the performance in a non-ELMo setting (denoted as GAIL-W2V) which utilizes the following embedding techniques to represent tokens in the input sentence.

- Token surface embeddings: For each unique token in the training set, we have a look-up dictionary for embeddings which is randomly initialized and updated in the training phase.
- Character-based embeddings: Each character also has a randomly initialized embedding, and will be fed into a token-level Bi-LSTM network, and the final output of this network will enrich the information of tokens.
- POS embeddings: We apply Part-of-Speech (POS) tagging on the sentences using Stanford CoreNLP tool [40]. The POS tags of the tokens also have a trainable look-up dictionary (embeddings).
- Pre-trained embeddings: We also acquire embeddings trained from a large and publicly available corpus. These embeddings preserve semantic information of the tokens and they are not updated in the training phase.

[④] We use pretrained version at https://www.tensorflow.org/hub/modules/google/elmo/2.

We concatenate these embeddings and feed them into the Bi-LSTM networks as demonstrated in Figures 2 and 4. To relieve over-fitting issues, we utilize dropout strategy on the input data during the training phase. We intentionally set "UNK" (unknown) masks, which hold entries in the look-up dictionaries of tokens, POS tags and characters. We randomly mask known tokens, POS tags and characters in the training sentences with the "UNK" mask. We also set an all-0 vector on Word2Vec embeddings of randomly selected tokens.

We tune the parameters according to the F1 score of argument role labeling. For Q-Learning, we set a discount factor $\gamma = 0.01$. For all RL tasks, we set exploration threshold $\varepsilon = 0.1$. We set all hidden layer sizes (including the ones on discriminators) and LSTM (for subsentence Bi-LSTM) cell memory sizes as 128. The dropout rate is 0.2. When optimizing the parameters in the neural networks, we use stochastic gradient descent (SGD) with momentum and the learning rates start from 0.02 (sequence labeling), 0.005 (argument labeling) and 0.001 (discriminators), and then the learning rate will decay every 5 epochs with exponential of 0.9; all momentum values are set as 0.9.

For the non-ELMo setting, we set 100 dimensions for token embeddings, 20 for POS embeddings, and 20 for character embeddings. For pre-trained embeddings, we train a 100-dimension Word2Vec [41] model from English Wikipedia articles (January 1, 2017), with all tokens preserved and a context window of five from both left and right.

We also implement an RL framework with fixed rewards of ± 5 as baseline with identical parameters as above. For sequence labeling (entity and event trigger detection task), we also set an additional reward value of -50 for B–I errors, namely, an I–label does not follow B– label with the same tag name (e.g., I–GPE follows B–PER). We use RL-W2V and RL-ELMo to denote these fixed-reward settings.

7.2 Results and Analysis

7.2.1 Entity Extraction Performance

We compare the performance of entity extraction (including named entities and nominal mentions) with the following state-of-the-art and high-performing approaches:

- JointIE [42]: A joint approach that extracts entities, relations, events and argument roles using structured prediction with rich local and global linguistic features.
- JointEntityEvent [6]: An approach that simultaneously extracts entities and arguments with document context.
- Tree-LSTM [43]: A Tree-LSTM based approach that extracts entities and relations.
- KBLSTM [8]: An LSTM-CRF (conditional random field) hybrid model that applies knowledge base information on sequence labeling.

From Table 1 we can conclude that our proposed method outperforms the other approaches, especially with an impressively high performance of recall. CRF-based models are applied on sequence labeling tasks because CRF can consider the label on previous token to avoid mistakes such as appending an I-GPE to a B-PER, but it neglects the information from the later tokens. Our proposed approach avoids the aforementioned mistakes by issuing strong penalties (negative reward with large absolute value); and the Q-values in our sequence labeling sub-framework also considers rewards for the later tokens, which significantly enhances our prediction performance.

Table 1. Entity extraction performance.

	P	R	F1
JointIE	85.2	76.9	80.8
JointEntityEvent	83.5	80.2	81.8
Tree-LSTM	82.9	83.9	83.4
KBLSTM	85.4	86.0	85.7
RL-W2V	82.0	86.1	84.0
RL-ELMo	83.1	87.0	85.0
GAIL-W2V	85.4	88.6	**86.9***
GAIL-ELMo	85.8	89.7	**87.1***

Note: * means statistically significant (p < 0.05 with Wilcoxon signed rank test) against KBLSTM [8].

7.2.2 Event Extraction Performance

For event extraction performance with system-predicted entities as argument candidates, besides [42] and [6] we compare our performance with:

- dbRNN [16]: an LSTM framework incorporating the dependency graph (dependency-bridge) information to detect event triggers and argument roles.

Table 2 demonstrates that the performance of our proposed framework is better than state-of-the-art approaches except lower F1 score on argument identification against [16]. [16] utilizes Stanford CoreNLP to detect the noun phrases and take the detected phrases as argument candidates, while our argument candidates come from system predicted entities and some entities may be missed. However, [16]'s approach misses entity type information, which causes many errors in argument role labeling task, whereas our argument candidates hold entity types, and our final role labeling performance is better than [16].

Our framework is also flexible to consume ground-truth (gold) annotation of entities as argument candidates. And we demonstrate the performance comparison with the following state-of-the-art approaches on the same setting besides [16]:

- JointIE-GT [4]: similar to [42], the only difference is that this approach detects arguments based on ground-truth entities.
- JRNN [13], an RNN-based approach which integrates local lexical features.

For this setting, we keep the identical parameters (including both trained and preset ones) and network structures which we use to report our performance in Tables 1 and 2, and we substitute system-predicted entity types and offsets with ground-truth counterparts.

Table 3 demonstrates that, without any further deliberate tuning, our proposed approach can still provide better performance.

Table 2. Performance comparison with state-of-the-art frameworks with system predicted entities.

Tasks Metric	Trigger Identification			Trigger Labeling			Argument Identification			Role Labeling		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
JointIE	-	-	-	65.6	61.0	63.2	-	-	-	60.5	39.6	47.9
JointEntityEvent	77.6	65.4	71.0	75.1	63.3	68.7	73.7	38.5	50.6	70.6	36.9	48.4
dbRNN	-	-	-	-	-	69.6	-	-	57.2	-	-	50.1
RL-W2V	73.9	64.8	69.0	69.9	62.1	65.8	58.5	48.2	52.9	53.4	44.7	48.6
RL-ELMo	74.1	65.6	69.6	70.4	62.2	66.0	57.6	47.2	51.9	54.2	43.7	48.4
GAIL-W2V	76.5	70.9	73.2	74.4	69.3	**71.8**	62.3	48.2	54.3	61.7	44.8	**51.9**
GAIL-ELMo	76.8	71.2	**73.9**	74.8	69.4	**72.0**	63.3	48.7	55.1	61.6	45.7	**52.4**

Table 3. Comparison (F1) with state-of-the-art frameworks on ground-truth (gold) entity as argument candidates.

Tasks	TI	TL	AI	RL
JointIE-GT	70.4	67.5	56.8	52.7
JRNN	71.9	69.3	62.8	55.4
dbRNN	-	71.9	67.7	58.7
RL-W2V	71.2	69.7	58.9	54.8
RL-ELMo	71.1	69.5	58.7	54.6
GAIL-W2V	**74.6**	**72.7**	**67.8**	**59.1**
GAIL-ELMo	**74.6**	**72.9**	**67.9**	**59.7**

Note: TI=Trigger Identification, TL=Trigger Labeling, AI=Argument Identification and RL=Role Labeling.

7.2.3 Merit of Dynamic Rewards

The statistical results in Tables 1, 2 and 3 demonstrate that dynamic rewards outperform the settings with fixed rewards. As presented in Section 5, fixed reward setting resembles classification methods with cross-entropy loss, which treat errors equally and do not incorporate much information from errors, and hence the performance is similar to some earlier approaches but does not outperform state-of-the-art.

For the instances with ambiguity, our dynamic reward function can provide more salient margins between correct and wrong labels. With the identical parameter set as aforementioned, reward for the wrong Die label is as lower as –8.27 while correct Execute label gains as high as 9.35. Figure 6 illustrates the curves of rewards with regard to epoch numbers. For easier instances, e.g., the trigger word "*arrested*" in "... *police have arrested* ..." have flatter reward values as 1.24 for Arrest-Jail, –1.53 for None or –1.37 for Attack, which are sufficient for correct labels.

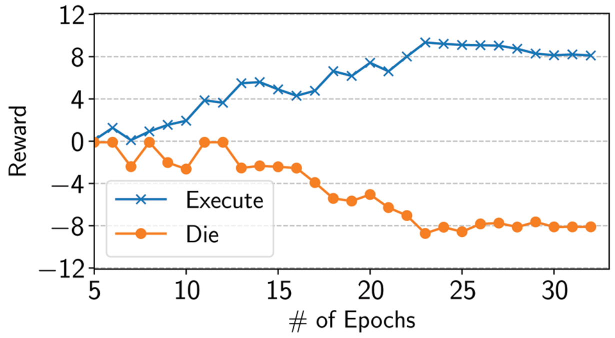


Figure 6. Change of rewards with regard to event type labels on the trigger "***death***" mentioned in Figure 1.

7.2.4 Impact from Pretrained Embeddings

Scores in Tables 1, 2 and 3 prove that non-ELMo settings already outperform state-of-the-art, which confirms the advantage and contribution of our GAIL framework. Moreover, in spite of insignificant drop in fixed reward setting, we agree that ELMo is a good replacement for a combination of word, character and POS embeddings. The only shortcoming according to our empirical practice is that ELMo takes huge amount of GPU memory and the training procedure is slow (even we do not update the pre-trained parameters during our training phase).

7.3 Remaining Errors

Losses of scores are mainly missed trigger words and arguments. For instance, the End-Position trigger "*sack*" is missed because it is considered informal to use the word to express an End-Position event, and there are no similar tokens in the training data or in pre-trained embeddings. Another example of error is due to informal expression, e.g., "*I miss him to* ***death***", where the "***death***" does not trigger any event, while our system makes a mistake by detecting it as a Die event. Since most training sentences are formal writing, expression from oral speeches which are usually informal may cause errors.

We also notice that there are some special errors which are caused by biased annotation. In the sentence "*Bush invites his 'good friend' Putin to his weekend '****retreat****' outside Camp David in Washington in September*", the FAC (facility) entity "***retreat***" is mistakenly labeled as a trigger word of Transport. In the training data, all the "***retreat***" tokens (or "***retreated***", "***retreating***", "***retreats***" are labeled as Transport;

however, this "***retreat***" means a facility which is "a quiet or secluded place for relaxation". We also notice that the reward value for FAC (correct label) is –2.73 and Transport (wrong label) 3.13, which contradicts the expectation that correct label comes with higher reward value. This error implies that our approach still requires a mixture of all possible labels so that it is able to acknowledge and explore the ambiguous and possible actions.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we propose an end-to-end entity and event extraction framework based on inverse reinforcement learning. Experiments have demonstrated that the performance benefits from dynamic reward values estimated from discriminators in a GAN, and we also demonstrate the performance of recent embedding work in the experiments. In the future, besides releasing the source code, we also will attempt to interpret the dynamics of these rewards with regard to the instances so that researchers and event extraction system developers are able to better understand and explore the algorithm and remaining challenges. Our future work also includes using cutting edge approaches such as BERT [44], and exploring joint model in order to alleviate impact from upstream errors in current pipelined framework.

ACKNOWLEDGEMENTS

This work was supported by the US Air Force No. FA8650-17-C-7715, DARPA AIDA Program No. FA8750-18-2-0014, and US ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Government. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

AUTHOR CONTRIBUTIONS

T. Zhang (zhangt13@rpi.edu) contributed to the design and implementation of the research. All authors, T. Zhang, H. Ji (jih@rpi.edu, corresponding author) and Avirup Sil (avi@us.ibm.com), contributed to the analysis of the results and to the writing of the manuscript.

REFERENCES

[1] C. Walker, S. Strassel, J. Medero, & K. Maeda. ACE 2005 multilingual training corpus. Philadelphia: Linguistic Data Consortium, 2006. isbn: 1-58563-376-3.

[2] Linguistic Data Consortium. ACE (Automatic Content Extraction) English Annotation Guidelines for events. Availableat: https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf.

[3] Z. Song, A. Bies, S. Strassel, T. Riese, J. Mott, J. Ellis, J. Wright, S. Kulick, N. Ryant, & X. Ma. From light to rich ERE: Annotation of entities, relations, and events. In: Proceedings of the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation, ACL, 2015, pp. 89–98. doi: 10.3115/v1/W15-0812.

[4] Q. Li, H. Ji, & L. Huang. Joint event extraction via structured prediction with global features. In: Proceedings of 2013 Annual Meeting of the Association for Computational Linguistics, ACL, 2013, pp. 73–82. Available at: http://aclweb.org/anthology/P/P13/P13-1008.pdf.

[5] T. Zhang, S. Whitehead, H. Zhang, H. Li, J. Ellis, L. Huang, W. Liu, H. Ji, & S.-F. Chang. Improving event extraction via multimodal integration. In: Proceedings of the 2017 ACM on Multimedia Conference, ACM, 2017, pp. 270–278. doi: 10.1145/3123266.3123294.

[6] B. Yang, & T. Mitchell. Joint extraction of events and entities within a document context. In: Proceedings of 2016 Annual Conference of the North American Chapter of the Association for Computational Linguistics, ACL, 2016, pp. 289–299. doi: 10.18653/v1/N16-1033.

[7] A. Judea, & M. Strube. Incremental global event extraction. In: The 26th International Conference on Computational Linguistics: Technical Papers, International Committee on Computational Linguistics, 2016, pp. 2279–2289. Available at: http://aclweb.org/anthology/C/C16/C16-1215.pdf.

[8] B. Yang, & T. Mitchell. Leveraging knowledge bases in LSTMs for improving machine reading. In: Proceedings of 2017 Annual Meeting of the Association for Computational Linguistics, ACL, 2017, pp. 1436–1446. doi: 10.18653/v1/P17-1132.

[9] S. Duan, R. He, & W. Zhao. Exploiting document level information to improve event detection via recurrent neural networks. In: Proceedings of 2017 International Joint Conference on Natural Language Processing (Poster and Demo Session), 2017. Available at: http://ijcnlp2017.org/site/page.aspx?pid=172&sid=1133&lang=en.

[10] Y. Chen, L. Xu, K. Liu, D. Zeng, & J. Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, ACM, 2015, pp. 167–176. doi: 10.3115/v1/P15-1017.

[11] T.H. Nguyen, & R. Grishman. Event detection and domain adaptation with convolutional neural networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), ACM, 2015, pp. 365–371. Available at: http://www.anthology.aclweb.org/P/P15/P15-2060.pdf.

[12] X. Feng, L. Huang, D. Tang, B. Qin, H. Ji, & T. Liu. A language independent neural network for event detection. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL, pp. 66–71. Available at: http://wing.comp.nus.edu.sg/~antho/P/P16/P16-2011.pdf.

[13] T.H. Nguyen, K. Cho, & R. Grishman. Joint event extraction via recurrent neural networks. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL, 2016, pp. 300–309. doi: 10.18653/v1/N16-1034.

[14] L. Huang, T. Cassidy, X. Feng, H. Ji, C.R. Voss, J. Han, & A. Sil. Liberal event extraction and event schema induction. In: Proceedings of 2016 Annual Meeting of the Association for Computational Linguistics, ACM, 2016, doi: 10.18653/v1/P16-1025.

[15] T.H. Nguyen, & R. Grishman. Graph convolutional networks with argument-aware pooling for event detection. In: Association for the Advancement of Artificial Intelligence 2018. Available at: http://ix.cs.uoregon.edu/~thien/pubs/graphConv.pdf

[16] L. Sha, F. Qian, S. Li, B. Chang, & Z. Sui. Jointly extracting event triggers and arguments by ependency-bridge RNN and tensorbased argument interaction. In: Association for the Advancement of Artificial Intelligence 2018. Available at: http://shalei120.github.io/docs/sha2018Joint.pdf.

[17] L. Huang, H. Ji, K. Cho, I. Dagan, S. Riedel, & C. Voss. Zero-shot transfer learning for event extraction. In: Proceedings of 2018 Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Available at: http://nlp.cs.rpi.edu/paper/zeroshot2017.pdf.

[18] Y. Hong, W. Zhou, J. Zhang, Q. Zhu, & G. Zhou. Self-regulation: Employing a generative adversarial network to improve event detection. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), ACL, 2018, pp. 515–526. Available at: http://aclweb.org/anthology/P18-1048.

[19] Y. Zhao, X. Jin, Y. Wang, & X. Cheng. Document embedding enhanced event detection with hierarchical and supervised attention. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL, pp. 414–419. Available at: http://aclweb.org/anthology/P18-2066.

[20] T.M. Nguyen, & T.H. Nguyen. One for all: Neural joint modeling of entities and events. arXiv preprint. arXiv:1812.00195, 2018.

[21] Y. Feng, H. Zhang, W. Hao, & G. Chen. Joint extraction of entities and relations using reinforcement learning and deep learning. Computational Intelligence and Neuroscience (2017), Article ID 7643065. doi: 10.1155/2017/7643065.

[22] H. Zhang, Y. Feng, W. Hao, G. Chen, & D. Jin. Relation extraction with deep reinforcement learning. IEICE Transactions on Information and Systems E100.D(8)(2017), 1893–1902. doi: 10.1587/transinf.2016EDP7450.

[23] H. Daumé, J. Langford, & D. Marcu. Search-based structured prediction. Machine learning 75(3) 2009, 297–325. doi: 10.1007/s10994-009-5106-x.

[24] S. Ross, G. Gordon, & D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of 2011 International Conference on Artificial Intelligence and Statistics. arXiv preprint. arXiv: arXiv:1011.0686, 2011.

[25] K.-W. Chang, A. Krishnamurthy, A. Agarwal, H. Daumé III, & J. Langford. Learning to search better than your teacher. In: Proceedings of the 32nd International Conference on Machine Learning, ICML, 2015, pp. 2058–2066. Available at: https://dl.acm.org/citation.cfm?id=3045337.

[26] P. Abbeel, & A.Y. Ng. Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the 21st International Conference on Machine Learning, ICML, 2004, pp. 1–8. doi: 10.1145/1015330.1015430.

[27] U. Syed, M. Bowling, & R.E. Schapire. Apprenticeship learning using linear programming. Proceedings of the 25th International Conference on Machine Learning, ICML, 2008, pp. 1032–1039. doi: 10.1145/1390156.1390286.

[28] B.D. Ziebart, A.L. Maas, J.A. Bagnell, & A.K. Dey. Maximum entropy inverse reinforcement learning. In: Proceedings of the 23rd National Conference on Artificial Intelligence, AAAI, 2008, pp. 1433–1438. Available at: https://dl.acm.org/citation.cfm?id=1620297.

[29] J. Ho, & S. Ermon. Generative adversarial imitation learning. In: 30th Conference on Neural Information Processing Systems (NIPS 2016). Available at: http://papers.nips.cc/paper/6391-generative-adversarial-imitation-learning.pdf.

[30] N. Baram, O. Anschel, I. Caspi, & S. Mannor. End-to-end differentiable adversarial imitation learning. In: Proceedings of the 34th International Conference on Machine Learning, ICML, 2017, pp. 390–399. Available at: http://proceedings.mlr.press/v70/baram17a.html.

[31] A. Vlachos, & M. Craven. Searchbased structured prediction applied to biomedical event extraction. In: Proceedings of 2011 Conference on Computational Natural Language Learning, ACL, 2011, pp. 49–57. Available at: https://dl.acm.org/citation.cfm?id=2018943.

[32] F. Maes, L.Denoyer, & P. Gallinari. 2007. Sequence labeling with reinforcement learning and ranking algorithms. In: Proceedings of the 18th European Conference on Machine Learning, Springer, pp. 648–657. doi: 10.1007/978-3-540-74958-5_64.

[33] S. Hochreiter, & J. Schmidhuber. Long ¨short-term memory. Neural Computation 9(8)(1997), pp. 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

[34] B. Bakker. Reinforcement learning with long short-term memory. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, MIT Press, pp. 1475–1482. Available at: https://dl.acm.org/citation.cfm?id=2980731.

[35] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, & L. Zettlemoyer. Deep contextualized word representations. In: Proceedings of 2018 Annual Conference of the North American Chapter of the Association for Computational Linguistics, ACL, 2018, pp. 2227–2237. doi: 10.18653/v1/N18-1202.

[36] R.S. Sutton, D.A. McAllester, S.P. Singh, & Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In: S.A. Solla,& T.K. Leen, & K.R. Müller (eds.) Advances in Neural Information Processing Systems 12 (NIPS 1999). Cambridge, MA: The MIT Press, 2000, pp. 1057–1063. Available at: http://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf.

[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, & Y. Bengio. Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, MIT Press, 2014, pp. 2672–2680. Available at: https://dl.acm.org/citation.cfm?id=2969125.

[38] R. Pasunuru, & M. Bansal. Reinforced video captioning with entailment rewards. arXiv preprint. arXiv:1708.02300, 2017.

[39] R. Pasunuru, & M. Bansal. Multireward reinforced summarization with saliency and entailment. arXiv preprint. arXiv:1804.06451, 2018.

[40] K. Toutanova, D. Klein, C.D. Manning, & Y. Singer. Feature-rich part-ofspeech tagging with a cyclic dependency network. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, ACL, 2003, pp. 173–180. doi: 10.3115/1073445.1073478.

[41] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, & J. Dean. Distributed representations of words and phrases and their compositionality. In: C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 26. Available at: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[42] Q. Li, H. Ji, Y. Hong, & S. Li. Constructing information networks using one single model. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), ACL, 2014, pp. 1846–1851. Available at: http://aclweb.org/anthology/D14-1198.

[43] M. Miwa, & M. Bansal. End-to-end relation extraction using LSTMs on sequences and tree structures. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL, 2016, pp. 1105–1116. Available at: http://aclweb.org/anthology/P16-1105.

[44] H.J. Devlin, M.-W. Chang, K. Lee, & K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint. arXiv:1810.04805, 2018.

AUTHOR BIOGRAPHY

Tongtao Zhang is a PhD candidate in the Computer Science Department, Rensselaer Polytechnic Institute and a member of BLENDER. He focuses on event extraction with multi-modal approaches, which encompass across domains such as natural language processing, computer vision and machine learning. He is attempting to fuse the techniques from these domains to pursue a more comprehensive knowledge base. Tongtao received MS degree from Department of Electrical Engineering, Columbia University, BS degree from Department of Applied Physics, Donghua University with "City's Graduate of Excellence" and BA from Department of German, Shanghai International Studies University.

Heng Ji is Edward P. Hamilton Development Chair Professor in Computer Science Department of Rensselaer Polytechnic Institute. She received her BA and MA in Computational Linguistics from Tsinghua University and her MS and PhD in Computer Science from New York University. Her research interests focus on natural language processing and its connections with data mining, Social Sciences and vision. She was selected as "Young Scientist" and a member of the Global Future Council on the Future of Computing by the World Economic Forum in 2016 and 2017. She received "AI's 10 to Watch" Award by *IEEE Intelligent Systems* in 2013, NSF CAREER award in 2009, Google Research Awards in 2009 and 2014, Sloan Junior Faculty Award in 2012, IBM Watson Faculty Award in 2012 and 2014, Bosch Research Awards in 2015, 2016 and 2017, PACLIC2012 Best Paper Runner-up, "Best of SDM2013" paper, and "Best of ICDM2013" paper. She is invited by the Secretary of the Air Force and AFRL to join Air Force Data Analytics Expert Panel to inform the Air Force Strategy 2030. She is the task leader of the US ARL projects on information fusion and knowledge networks construction. She led the Tinker Bell team that consists of seven universities under DARPA DEFT program. She coordinated the NIST TAC Knowledge Base Population task since 2010, served as the Program Committee Chair of NAACL2018, NLP-NABD2018, NLPCC2015 and CSCKG2016, ACL2017 Demo Co-Chair, the Information Extraction area chair for NAACL2012, ACL2013, EMNLP2013, NLPCC2014, EMNLP2015, NAACL2016, ACL2016 and NAACL2019, senior information extraction area chair of ACL2019, the vice Program Committee Chair for IEEE/WIC/ACM WI2013 and CCL2015, Content Analysis Track Chair of WWW2015, and the Financial Chair of IJCAI2016.

Dr. Avirup Sil is a Research Scientist in the Information Extraction and natural language processing (NLP) group at IBM Research AI. He is also the Chair of the NLP professional community of IBM. Currently, he is working on industry scale NLP and deep learning algorithms. His work is mainly on information extraction: entity recognition and linking and relation extraction. Currently, he is working on Question Answering algorithms by attaching world knowledge to systems. He is a senior program committee member for major Computational Linguistics conferences including being an Area Chair multiple times. Avirup finished his PhD in Computer Science under the supervision of his thesis advisor Alexander Yates. He also worked on temporal information extraction in the Machine Learning Group at Microsoft Research, Redmond, managed by Chris Burges and John Platt. His mentor was Silviu Cucerzan. He has more than 12 US patents filed all in the area of artificial intelligence and its applications in various spheres.