Joint Entity and Event Extraction with Generative Adversarial Imitation Learning

Tongtao Zhang[†], **Heng Ji**[†] **and Avirup Sil**[*]
[†]Computer Science Department, Rensselaer Polytechnic Institute
[*]IBM Research AI
{zhangt13, jih}@rpi.edu and {avi@us.ibm.com}

Abstract

We propose a new framework for entity and event extraction based on generative adversarial imitation learning – an inverse reinforcement learning method using 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 `Execute` event ("*death*") involving the person entity "*Masih*". Most event extraction research has been in the context of the 2005 NIST Automatic Content Extraction (ACE) sentence-level event mention task (Walker et al., 2006), which also provides the standard corpus. The annotation guideline of the ACE program defines an event as a specific occurrence of something that happens involving participants, often described as a change of state. More recently, the 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 (Song et al., 2015), introducing rich features of multiple categories (Li et al., 2013; Zhang et al., 2017b), incorporating relevant information within or beyond context (Yang and Mitchell, 2016; Judea and Strube, 2016; Yang and Mitchell, 2017; Duan et al., 2017) and adopting neural network frameworks (Chen et al., 2015; Nguyen and Grishman, 2015; Feng et al., 2016; Nguyen et al., 2016; Huang et al., 2016; Nguyen and Grishman, 2018; Sha et al., 2018; Huang et al., 2018; Hong et al., 2018; Zhao et al., 2018; Nguyen and Nguyen, 2018).

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 error in the former instance.

Such 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 treat 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

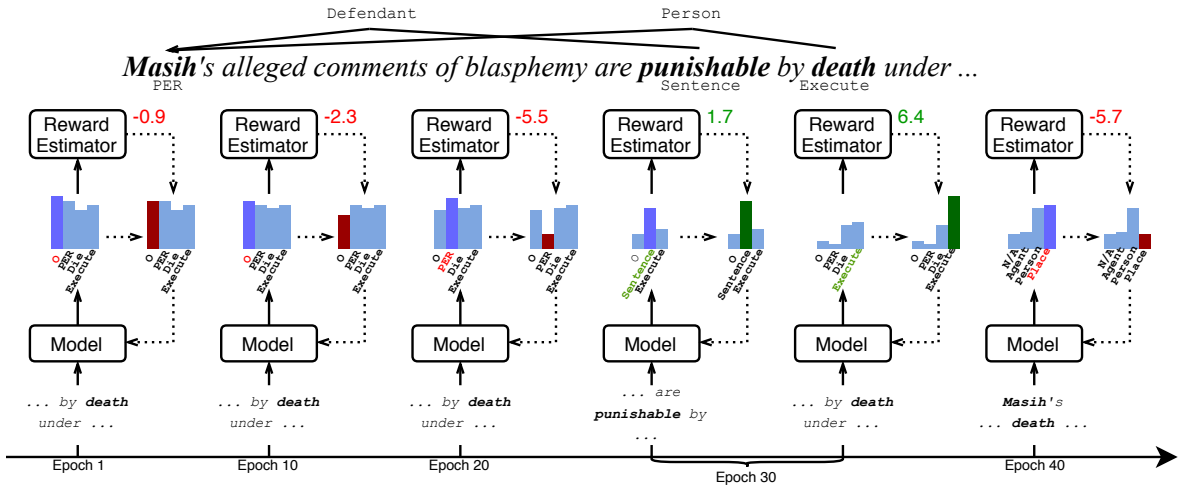

Figure 1: Our framework includes a reward estimator based on GAN to issue dynamic rewards with regard to the labels (actions) committed by event extractor (agent). The reward estimator is trained upon the difference between the labels from ground truth (expert) and extractor (agent). If the extractor repeatedly misses `Execute` label for "*death*", the penalty (negative reward values) is strengthened; if the extractor make 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.

models robust.

In this paper, to combat the problems of previous approaches towards 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 generative adversarial networks (GAN) to estimate the reward values. Discriminators ensures 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 reinforcement learning 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 reinforcement learning propelled by GAN, we demonstrate that a dynamic reward function ensures more optimal performance in a complicated RL task.

2 Task and Term Preliminaries

In this paper we follow the schema of Automatic Content Extraction (ACE) (Walker et al., 2006) 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 7 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.

[1] https://catalog.ldc.upenn.edu/LDC2006T06

For broader readers who might not be familiar with reinforcement learning, we briefly introduce by 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 follow 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 3.1); meanwhile, the extractor determines the relations between triggers and entities – *argument roles* with policy gradient (Section 3.2). During the training epochs, GANs estimate rewards which stimulate the extractor to pursue the most optimal joint model (Section 4).

3 Framework and Approach

3.1 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 reinforcement learning is a well-suited method (Maes et al., 2007).

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 tth token in the sentence, it observes information from the environment and its previous action a_{t-1} as its current *state* s_t; 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 token's context embedding $\boldsymbol{v}_t$, which is usually acquired from Bi-LSTM (Hochreiter and Schmidhuber, 1997) 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

$$s_t =< \boldsymbol{v}_t, a_{t-1} > . \quad (1)$$

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

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

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

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

Equation 2 and 3 suggest that an RNN-based framework which consumes current input and previous inputs and outputs can be adopted, and we use a unidirectional LSTM as (Bakker, 2002). 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^*).

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

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 $Q'_{sl}(s_t, a_t)$:

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

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.

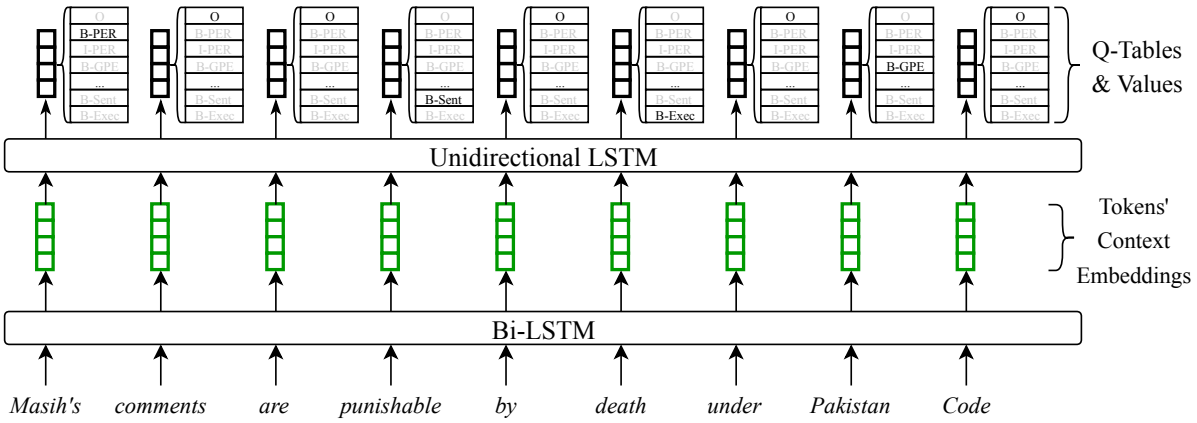

Figure 2: A pipeline from input sentence to sequence labels mentioned in Section 3.1. Q-table and values for each current step is calculated using the unidirectional LSTM based on context embeddings of current and previous tokens as well as Q-tables and values from previous steps. Context embeddings are calculated using Bi-LSTM from local token embeddings. Pre-trained embeddings based on Bi-LSTM such as ELMo (Peters et al., 2018) are also good candidates for context embeddings.

Q-Table at *m*th Epoch with a wrong 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
...	...	...

Back propagation and weight update

Q-Table at *n*th Epoch with a correct label

	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 wrong label is penalized while correct one is reinforced.

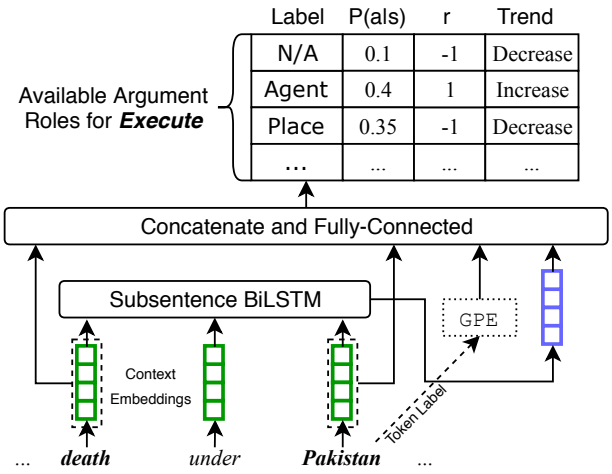

Figure 4: The extractor combines context embeddings of the trigger and entity, as well as a one-hot vector that represents entity type and Bi-LSTM output of sub-sentence between the trigger and argument. The column "trend" denotes the changes of $P(a_{tr,ar}|s_{tr,ar})$ after policy gradient optimization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

3.2 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 – $\boldsymbol{v}_{t_{tr}}$ and $\boldsymbol{v}_{t_{ar}}$ 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 determine the available argument role labels, e.g., an `Attack` event never has `Adjudicator` arguments as `Sentence` events. With these notations we have:

$$s_{tr,ar} =< \boldsymbol{v}_{t_{tr}}, \boldsymbol{v}_{t_{ar}}, a_{t_{tr}}, a_{t_{ar}}, \boldsymbol{f}_{ss} >, \quad (6)$$

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:

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

where $\boldsymbol{f}_{tr,ar}$ represents a mapping function whose output sizes is determined by the trigger event type $a_{t_{tr}}$. e.g., `Attack` event has 5 – `Attacker`, `Target`, `Instrument`, `Place` and `Not-a-role` labels and the mapping function for `Attack` event contains a fully-connected layer with output size of 5.

And we determine the role with

$$\hat{a}_{tr,ar} = \arg\max_{a_{tr,ar}} Q_{tr,ar}(s_{tr,ar}, a_{tr,ar}). \quad (8)$$

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 (Sutton et al., 2000) 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:

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

To update the parameters, we minimize loss function

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

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 $P(a_{tr,ar}|s_{tr,ar})$.

3.3 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 constrain the R_t but they still need additional estimation and bad estimation will introduce new risk. 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, 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 label may make the updated Q-value smaller than those wrong ones.

4 Generative Adversarial Imitation Learning

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

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

and typically we have $c_1 > c_2$.

This strategy makes RL-based approach no difference from classification approaches with cross-entropy in terms of "treating wrong labels equally" as discussed in introductory section. Moreover, recent RL approaches on relation extraction (Zhang et al., 2017a; Feng et al., 2017) 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 2 or 3 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 "*U.S. 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.

$$\mathbb{E}_{\pi_E}[R(s,a)] \geq \mathbb{E}_{\pi_A}[R(s,a)] \tag{12}$$

This mechanism is Inverse Reinforcement Learning (Abbeel and Ng, 2004), which estimates the reward first in an RL framework.

Equation 12 reveals a scenario of adversary between ground truth and extractor and Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016), which is based on GAN (Goodfellow et al., 2014), 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 provide lower rewards for other/wrong labels.

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

$$\mathbb{E}_{\pi_E}[D(s,a_E)] \geq \mathbb{E}_{\pi_A}[D(s,a_A)]. \tag{13}$$

where s, a_E and a_A are input 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, 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 (1 for sequence labeling, 33 for event argument role labeling with regard to 33 event types). Every discriminator consists of 2 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:

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

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:

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

During the training process, after we feed the neural network mentioned in Section 3.1 and 3.2 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 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 values.

Figure 5 illustrates how we utilize GAN for reward estimation.

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

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

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

5 Exploration

In training phase, the extractor selects labels according to the rankings of Q-values in Equa-

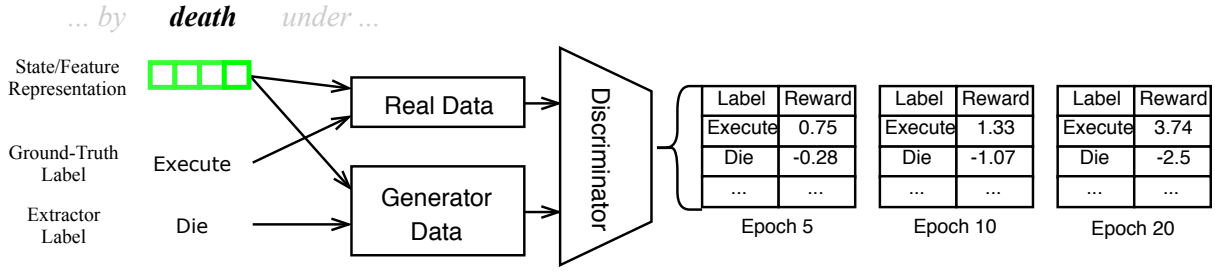

Figure 5: An illustrative example of the GAN structure in sequence labeling scenario (argument role labeling scenario has the identical frameworks except vector dimensions).As introduced in Section 4, the "real data" in the original GAN is replaced by feature/state representation (Equation 1, or Equation 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 $[0, 1]$.

tion 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 $\epsilon \in [0, 1)$ and uniformly sample a number $\rho \in [0, 1]$ before the extractor commits a label for an instance:

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

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 (Pasunuru and Bansal, 2017, 2018), 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.

6 Experiments

6.1 Experiment Setup

To evaluate the performance with our proposed approach, we utilize ACE2005 documents. To align with state-of-the-art frameworks such as (Nguyen et al., 2016; Sha et al., 2018), we exclude informal documents from `cts` (Conversational Telephone Speech) and `un` (UseNet), 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 SOTA 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] (Peters et al., 2018). Because ELMo is delivered with built-in Bi-LSTMs, we treat ELMo embedding as context embeddings in Figure 2 and 4. We use GAIL-ELMo in the tables to denote the setting.

Moreover, in order to disentangle the contribu-

[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.

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

tion 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, the final output of this network will enrich the information of token.
- POS embeddings: We apply Part-of-Speech (POS) tagging on the sentences using Stanford CoreNLP tool (Toutanova et al., 2003). 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.

We concatenate these embeddings and feed them into the Bi-LSTM networks as demonstrated in Figure 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 "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 $\epsilon = 0.1$. 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 SGD with Momentum and the learning rates start from 0.02 (sequence labeling), 0.005 (argument labeling) and 0.001 (discriminators), 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 (Mikolov et al., 2013) model from English Wikipedia articles (January 1st, 2017), with all tokens preserved and a context window of 5 from both left and right.

We also implemented 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.

6.2 Results and Analysis

6.2.1 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 (Li et al., 2014): a joint approach that extracts entities, relations, events and argument roles using structured prediction with rich local and global linguistic features.
- JointEntityEvent (Yang and Mitchell, 2016): an approach that simultaneously extracts entities and arguments with document context.
- Tree-LSTM (Miwa and Bansal, 2016): a Tree-LSTM based approach that extracts entities and relations.
- KBLSTM (Yang and Mitchell, 2017): an LSTM-CRF 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.

	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***

Table 1: Entity extraction performance. *: statistically significant ($p < 0.05$ with Wilcoxon signed rank test) against KBLSTM (Yang and Mitchell, 2017)

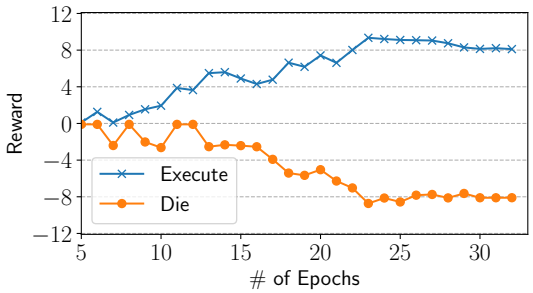

Figure 6: Change of rewards w.r.t. event type labels on the trigger "***death***" mentioned in Figure 1.

6.2.2 Event Extraction Performance

For event extraction performance with system-predicted entities as argument candidates, besides (Li et al., 2014) and (Yang and Mitchell, 2016) we compare our performance with:

- dbRNN (Sha et al., 2018): 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 (Sha et al., 2018). (Sha et al., 2018) 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, (Sha et al., 2018)'s approach misses entity type information, which cause many errors in argument role labeling task, whereas our argument candidates hold entity types, and our final role labeling performance is better than (Sha et al., 2018).

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 (Sha et al., 2018):

- JointIE-GT (Li et al., 2013): similar to (Li et al., 2014), the only difference is that this approach detects arguments based on ground-truth entities.
- JRNN (Nguyen et al., 2016), 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 used to report our performance in Table 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.

6.2.3 Merit of dynamic rewards

The statistical results in Table 1, 2 and 3 demonstrate that dynamic rewards outperforms the settings with fixed rewards. As presented in Section 4, fixed reward setting resembles classification methods with cross-entropy loss, which treat errors equally and do not incorporate much information from errors, 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 simpler cases, *e.g.*, "... *submitted his* ***resignation*** ...", we have flatter rewards as 2.74 for `End-Position`, -1.33 for `None` or -1.67 for `Meet`, which are sufficient to commit correct labels.

6.2.4 Impact from Pretrained Embeddings

Scores in Table 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

Tasks	Trigger Identification			Trigger Labeling			Argument Identification			Role Labeling		
Metric	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 2: Performance comparison with state-of-the-art frameworks with system predicted entities.

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**

Table 3: Comparison (F1) with State-of-the-Art frameworks on ground-truth (gold) entity as argument candidates. TI=Trigger Identification, TL=Trigger Labeling, AI=Argument Identification, RL=Role Labeling

and the training procedure is slow (even we do not update the pre-trained parameters during our training phase).

6.3 Remaining Errors

Losses of scores are mainly missed trigger words and arguments. For example, the `Meet` trigger "***pow-wow***" is missed because it is rarely used to describe a formal political meeting; and there is no token with similar surface form – which can be recovered using character embedding or character information in ELMo setting – in the training data. 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 observe some special erroneous cases due to fully biased annotation. In the sentence "***Bombers*** *have also hit targets ...*", the entity "***bombers***" is mistakenly classified as the `Attacker` argument of the `Attack` event triggered by the word "***hit***". Here the "***bombers***" refers to aircraft and is considered as a `VEH` (Vehicle) entity, and should be an `Instrument` in the `Attack` event, while "***bombers***" entities in the training data are annotated as `Person` (who detonates bombs), which are never `Instrument`. From the perspective of the reward estimator, the reward for wrong `Attacker` is 3.89 while the correct label `Instrument` has -2.3 – which is totally reversed and unexpected. This illustrates how the system fails on a fully biased annotation. This is an ambiguous case, however, it does not compromise our claim on the merit of our proposed framework against ambiguous errors, because our proposed framework still requires a mixture of different labels to acknowledge ambiguity.

7 Related Work

One of the recent event extraction approaches mentioned in the introductory section (Hong et al., 2018) utilizes 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 assess 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 (Feng et al., 2017; Zhang et al., 2017a) 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(Daumé et al., 2009; Ross et al., 2011; Chang et al., 2015) attempt to map the states to expert actions by following demonstration, which resembles supervised learning, while inverse reinforcement learning (Abbeel and Ng, 2004; Syed et al., 2008; Ziebart et al., 2008; Ho and Ermon, 2016; Baram et al., 2017) estimates the rewards first and

apply the rewards to RL. (Vlachos and Craven, 2011) is an imitation learning application on biomedical event extraction, and there is no reward estimator used. We humbly recognize our work as inverse reinforcement learning approach although "GAIL" is named after imitation learning.

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 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 dynamic 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 (Devlin et al., 2018), and exploring joint model in order to alleviate impact from upstream errors in current pipelined framework.

Acknowledgments

This work was supported by the U.S. Air Force No. FA8650-17-C-7715, DARPA AIDA Program No. FA8750-18-2-0014, and U.S. 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 U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of International Conference on Machine Learning 2004*.

Bram Bakker. 2002. Reinforcement learning with long short-term memory. In *Advances in neural information processing systems*, pages 1475–1482.

Nir Baram, Oron Anschel, Itai Caspi, and Shie Mannor. 2017. End-to-end differentiable adversarial imitation learning. In *Proceedings of International Conference on Machine Learning 2017*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume III, and John Langford. 2015. Learning to search better than your teacher.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, Jun Zhao, et al. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of 2015 Annual Meeting of the Association for Computational Linguistics*.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

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

Shaoyang Duan, Ruifang He, and Wenli Zhao. 2017. Exploiting document level information to improve event detection via recurrent neural networks. In *Proceedings of 2017 International Joint Conference on Natural Language Processing*.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Bing Qin, Heng Ji, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceddings of 2016 Annual Meeting of the Association for Computational Linguistics*.

Yuntian Feng, Hongjun Zhang, Wenning Hao, and Gang Chen. 2017. Joint extraction of entities and relations using reinforcement learning and deep learning. *Computational intelligence and neuroscience*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Yu Hong, Wenxuan Zhou, jingli, Guodong Zhou, and Qiaoming Zhu. 2018. Self-regulation: Employing a generative adversarial network to improve event detection. In *Proceedings of 2018 Annual Meeting of the Association for Computational Linguistics*.

Lifu Huang, T Cassidy, X Feng, H Ji, CR Voss, J Han, and A Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of 2016 Annual Meeting of the Association for Computational Linguistics*.

Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings of 2018 Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers).

Alex Judea and Michael Strube. 2016. Incremental global event extraction. In *Proceedings of 2016 International Conference on Computational Linguistics*.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of 2014 Conference on Empirical Methods on Natural Language Processing*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of 2013 Annual Meeting of the Association for Computational Linguistics*.

Francis Maes, Ludovic Denoyer, and Patrick Gallinari. 2007. Sequence labeling with reinforcement learning and ranking algorithms. In *Proceedings of 2007 European Conference on Machine Learning*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of 2016 Annual Meeting of the Association for Computational Linguistics*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of 2016 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of 2015 Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

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

Trung Minh Nguyen and Thien Huu Nguyen. 2018. One for all: Neural joint modeling of entities and events. *arXiv preprint arXiv:1812.00195*.

Ramakanth Pasunuru and Mohit Bansal. 2017. Reinforced video captioning with entailment rewards. *arXiv preprint arXiv:1708.02300*.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. *arXiv preprint arXiv:1804.06451*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of 2018 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of 2011 international conference on artificial intelligence and statistics*.

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

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: annotation of entities, relations, and events. In *Proceedings of Workshop on EVENTS: Definition, Detection, Coreference, and Representation, workshop at the North American Chapter of the Association for Computational Linguistics Conference*.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*.

Umar Syed, Michael Bowling, and Robert E Schapire. 2008. Apprenticeship learning using linear programming. In *Proceedings of 2008 international conference on Machine learning*.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech 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-Volume 1*. Association for Computational Linguistics.

Andreas Vlachos and Mark Craven. 2011. Search-based structured prediction applied to biomedical event extraction. In *Proceedings of 2011 Conference on Computational Natural Language Learning*.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

Bishan Yang and Tom Mitchell. 2016. 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*.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of 2017 Annual Meeting of the Association for Computational Linguistics*.

Hongjun Zhang, Yuntian Feng, Wenning Hao, Gang Chen, and Dawei Jin. 2017a. Relation extraction with deep reinforcement learning. *IEICE TRANSACTIONS on Information and Systems*, 100(8).

Tongtao Zhang, Spencer Whitehead, Hanwang Zhang, Hongzhi Li, Joseph Ellis, Lifu Huang, Wei Liu, Heng Ji, and Shih-Fu Chang. 2017b. Improving event extraction via multimodal integration. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM.

Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2018. Document embedding enhanced event detection with hierarchical and supervised attention. In *Proceedings of 2018 Annual Meeting of the Association for Computational Linguistics*.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning. In *AAAI 2008*.