One for All: Neural Joint Modeling of Entities and Events

Trung Minh Nguyen[*]
Alt Inc.
8F, Higashi-Kanda 3-1-2, Chiyoda-ku
Tokyo, 101-0031, Japan
nguyen.minh.trung@alt.ai

Thien Huu Nguyen[*]
Department of Computer and Information Science
University of Oregon
Eugene, Oregon 97403, USA
thien@cs.uoregon.edu

Abstract

The previous work for event extraction has mainly focused on the predictions for event triggers and argument roles, treating entity mentions as being provided by human annotators. This is unrealistic as entity mentions are usually predicted by some existing toolkits whose errors might be propagated to the event trigger and argument role recognition. Few of the recent work has addressed this problem by jointly predicting entity mentions, event triggers and arguments. However, such work is limited to using discrete engineering features to represent contextual information for the individual tasks and their interactions. In this work, we propose a novel model to jointly perform predictions for entity mentions, event triggers and arguments based on the shared hidden representations from deep learning. The experiments demonstrate the benefits of the proposed method, leading to the state-of-the-art performance for event extraction.

Introduction

An important problem of information extraction in natural language processing (NLP) is event extraction (EE): understanding how events are presented in text and developing techniques to recognize such events. We follow the definition of events in the annotation guideline designed for the ACE 2005 dataset[1]: an event is triggered by some words in the sentence with which several entities are associated to play different roles in the event.

EE is a challenging problem as it is a composition of three subtasks corresponding to different aspects of the event definition. In particular, the first subtask concerns the extraction of entity mentions appearing in the sentences (Entity Mention Detection - EMD) while the second subtask needs to identify the event trigger words (Event Detection - ED). Finally, in the third subtask, the relationships between the detected entity mentions and trigger words in the sentences should be recognized to reflect the roles of the entity mentions in the events (Argument Role Prediction - ARP). We call the three subtasks ordered by "EMD → ED → ARP" the EE pipeline for convenience. For instance, consider the following sentence taken from the ACE 2005 dataset:

[1]https://www.ldc.upenn.edu/collaborations/past-projects/ace

Another a-10 warthog was hit today.

In this sentence, an EMD system needs to recognize "*a-10 warthog*" as an entity mention of type "*VEHICLE*" and "*today*" as a time expression. For ED, the systems should be able to realize that "*hit*" is a trigger word for an event of type "*Attack*". Finally, for ARP, the systems are supposed to identify "*a-10 warthog*" as playing the "*Target*" role in the "*Attack*" event and "*today*" as the event's time.

A large portion of the prior work on EE has taken a simplified approach that only focuses on one or two specific subtasks, either assuming the manual/golden annotation for the other subtasks or simply ignoring them (i.e, the pipelined approach) (Li, Ji, and Huang 2013; Chen et al. 2015; Nguyen, Cho, and Grishman 2016a). One of the major issues with this approach is the error propagation in which the error from the earlier subtask is inherited and magnified in the later subtasks, causing the poor performance of those later subtasks (Li, Ji, and Huang 2013). In addition, the pipelined approach for EE does not have any mechanisms to capture the dependencies and interactions among the three subtasks so the later subtasks in the pipeline can interfere with and improve the decision process for the earlier subtasks. The earlier subtasks, on the other hand, can only communicate with the later subtasks via the discrete outputs, and fail to pass deeper information to the later stages to potentially improve the overall performance. Consider an EE system where EMD is done separately from ED and ARP as an example. In this system, the EMD module would work on its own and the ED and ARP modules have no way to correct the mistake made earlier by the EMD module. At the same time, it is very usual that the EMD module can only provide the ED and ARP modules with the boundary and types of the detected entity mentions. Such deeper information as the hidden contextual representations or the more fine-grained semantic classification of the entity mentions cannot be passed to or affect the ED and ARP modules. This would cause the inefficiency to use the information among the subtasks and result in the poor performance for EE.

It is thus appealing to design a single system to simultaneously model the three EE subtasks to avoid the aforementioned issues of the pipelined approach. However, due to the complexity in modeling, there has been only few works in the literature to study this joint modeling approach for EE. The major prior works in this direction for the ACE 2005

dataset involve (Li et al. 2014b), (Judea and Strube 2016) and (Yang and Mitchell 2016). Although these studies address the issues associated with the separate approach to some extent, they share the same limitation in which binary features (i.e, lexical words, dependency paths, etc.) are the main tools to capture the context for the individual subtasks and the dependencies/interactions among them. The major issue of those binary features is the inability to generalize over unseen words/features (due to the hard matches of binary features) and the limited expressiveness to encode the effective hidden structures for EE (Nguyen, Cho, and Grishman 2016a). Specifically, such binary representations cannot take advantages of the deep learning (DL) models with the shared hidden representations across different stages, a useful mechanism to enable the communications among the subtasks for EE demonstrated in (Nguyen, Cho, and Grishman 2016a).

In order to overcome the issues of such prior works for EE, in this paper, we propose a single deep learning model to jointly solve the three subtasks EMD, ED and ARP of EE. In particular, we employ a bidirectional recurrent neural network (RNN) to induce the shared hidden representations for the words in the sentence, over which the predictions for all the three subtasks EMD, ED and ARP are made. On the one hand, the bidirectional RNN helps to induce effective underlying structures via real-valued representations for the EE subtasks and mitigate the issue of hard matches for binary features. On the other hand, the shared hidden representations for the three subtasks enable the knowledge sharing across the subtasks so the hidden dependencies/interactions of the subtasks can be exploited to improve the EE performance.

We conduct extensive experiments to evaluate the effectiveness of the proposed model. The experiments demonstrate the benefits of joint modeling with deep learning for the three subtasks of EE over the traditional baselines, yielding the state-of-the-art performance on the long-standing and widely-used dataset ACE 2005. To the best of our knowledge, this is the first work to jointly model EMD, ED and ARP with deep learning.

Related Work

The early work on EE has mainly focused on the pipelined approach that performs the subtasks for EE separately and heavily relies on feature engineering to extract a diversity of features (Grishman, Westbrook, and Meyers 2005; Ahn 2006; Ji and Grishman 2008; Gupta and Ji 2009; Patwardhan and Riloff 2009; Liao and Grishman 2010; 2011; Hong et al. 2011; McClosky, Surdeanu, and Manning 2011; Huang and Riloff 2012; Miwa et al. 2014; Li et al. 2015). Some recent work has developed joint inference models for ED and ARP to address the error propagation issue in the pipelined approach. Those work exploits different structured prediction methods, including Markov Logic Networks (Riedel et al. 2009; Poon and Vanderwende 2010; Venugopal et al. 2014), Structured Perceptron (Li, Ji, and Huang 2013; Li et al. 2014b; Judea and Strube 2016) and Dual Decomposition (Riedel et al. 2009; Riedel and McCallum 2011a; 2011b). The closest work to ours is (Yang and

Mitchell 2016) that attempts to jointly model EMD, ED and ARP for EE. However, this work needs to find entity mentions and event trigger candidate separately. It also does not employ shared hidden feature representations as we do in this work with DL.

Deep learning has been shown to be very successful for EE recently. Most of the early work in this direction has also followed the pipelined approach (Nguyen and Grishman 2015b; Chen et al. 2015; Nguyen and Grishman 2016d; Chen et al. 2017; Liu et al. 2017; Nguyen and Grishman 2018a; Liu et al. 2018; Huang et al. 2018; Lu and Nguyen 2018) while some work on joint inference for EE has also been introduced (Nguyen, Cho, and Grishman 2016a; Sha et al. 2018). However, these studies are limited to the joint modeling of ED and ARP only.

Model

We propose a joint model for the three subtasks of EE (i.e, EMD, ED and ARP) at the sentence level. Let $W = w_1, w_2, \ldots, w_n$ be a sentence with n as the number of words/tokens and w_i as the i-th token. In order to solve the EMD problem, we cast it as a sequence labeling problem that attempts to assign a label e_i for every word w_i in W. The result is a label sequence $E = e_1, e_2, \ldots, e_n$ for W that can be used to reveal the boundary of the entity mentions and their entity types in the sentence. We apply the BIO annotation schema to generate the BIO labels for the words in the sentences[2].

Regarding the ED task for triggers, we follow the prior works (Li, Ji, and Huang 2013; Nguyen, Cho, and Grishman 2016a; Sha et al. 2018) to assume event triggers to be only single words/tokens in the sentences. This essentially leads to a word classification problem for every word in the sentences in which we need to predict an event type t_i for $w_i \in W$ (t_i can be "*Other*" to indicate the word w_i is not triggering any event of interest). The sequence of event type labels for the words in W is denoted by $T = t_1, t_2, \ldots, t_n$.

Finally, for event arguments, we need to recognize the entity mentions that are arguments for the event mentions appearing in W. However, as the event mentions and triggers are not provided in advanced in our setting, we essentially need to predict the argument role label for every pair of entity mention candidates and trigger candidates in the sentence. We choose the indexes of the beginning tokens of the entity mentions as the single anchors for the entity mentions. This translates into an argument role label matrix $A = (a_{ij})_{i,j=1}^{n}$ to encode the argument information of the events in W. In this square matrix, a_{ij} is set to "*Other*" if any of the following conditions is satisfied: (i) $i = j$, (ii) w_i is not a trigger word for any events in W, and (iii) w_j is not the beginning token of any entity mentions in W. Otherwise, if all the conditions are not satisfied, a_{ij} will be the argument role label that the entity mention with the beginning token w_j has in the event mention associated with the trigger word w_i. For convenience, we denote a_i as the i-th

---

[2]Following (Yang and Mitchell 2016), we consider values and time expressions as two additional entity types for prediction in the ACE 2005 dataset.
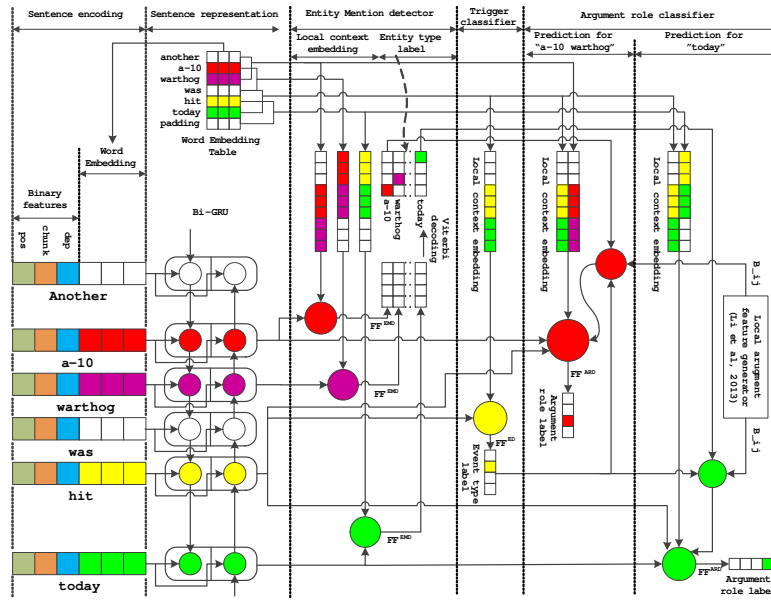
Figure 1: The joint EE model for the three subtasks with the input sentence "*Another a-10 warthog was hit today*" with local context window $u = 1$. Red and violet correspond to the beginning and last tokens of the entity mention "*a-10 warthog*" while green corresponds to the time "*today*". The trigger candidate "*hit*" at the current token is associated with yellow.

row in the matrix A. Given this encoding schema, the goal of the ARP module in our model is to predict the labels for the elements a_{ij} of A using the context specific to the tokens w_i and w_j.

The overall architecture of the joint model for EE in this work involves five components : i.e, sentence encoding, sentence representation, entity mention detector, trigger classifier and argument role classifier. These components are chained in an order as demonstrated in Figure 1 (from left to right). The first two steps help to transform the input sentence W into a hidden representation while the last three steps consume this hidden representation to make predictions for the three subtasks EMD, ED and ARP of EE.

Sentence Encoding

In the first component of *sentence encoding*, every word $w_i \in W$ is transformed into a vector x_i using the concatenation of the following vectors:

1. The pre-trained word embedding d_i of x_i (Mikolov et al. 2013a). We update the pre-trained word embeddings during the training process.

2. The binary vectors to capture the POS, chunk, and dependency information for w_i in W (Nguyen, Cho, and Grishman 2016a). In particular, we first run a POS tagger, chunker and dependency parser over the input sentence W. The results are then used to collect the POS tag, chunking tag (with the BIO annotation schema) and the surrounding dependency relations in the parse tree for w_i. Finally, we create one-hot vectors to represent the POS tag and chunking tag for w_e as well as a binary vector to indicate which dependency relations surround w_i in the dependency tree.

Sentence Representation

After the sentence encoding step, the input sentence W becomes a sequence of vectors $X = x_1, x_2, \ldots, x_n$. In the *sentence representation* component, the sequence of vectors X is fed into a bidirectional recurrent neural network (Hochreiter and Schmidhuber 1997; Cho et al. 2014) to generate the hidden vector sequence $H = h_1, h_2, \ldots, h_n$ for every word in the W. We employ the Gated Recurrent Units (GRU) (Cho et al. 2014) to implement the RNN model in this work. It has been shown that the hidden vector sequence $h_1, h_2, \ldots, h_n$ encodes rich contextual information of the whole sentence in each of the hidden vector h_i for EE (Nguyen, Cho, and Grishman 2016a). It is important to note that we utilize H as the shared representation to make the predictions for all the following components for EMD, ED and ARP. This enables the communications and facilitates the knowledge transfer among the three subtasks.

In order to decode W for EE, our goal is to predict the label variables in E, T and A jointly. Formally, this amounts to estimating the joint probability $P(A, T, E|W)$ for the input sentence W. In this work, we decompose this probability as follows to guide our design of the model architecture:

$$
\begin{aligned}
P(A, T, E|W) &= P(E|W) \times P(A, T|E, W) \\
&= P(E|W) \times P(a_1, t_1|E, W) \\
&\quad \times P(a_2, t_2|E, W, a_{<2}, t_{<2}) \\
&\quad \cdots \\
&\quad \times P(a_n, t_n|E, W, a_{<n}, t_{<n})
\end{aligned}
$$

where a_i denote the i-th row in the argument role label matrix A and $t_{<i} = t_1, t_2, \ldots, t_{i-1}$, $a_{<i} = a_1, a_2, \ldots, a_{i-1}$.

Based on this decomposition, we would first predict the entity type label e_i for every word in the sentence (i.e, com-

puting $P(E|W)$ for EMD) in the *entity mention detector* component. Afterward, the sentence is scanned from left to right for which the probability $P(a_i, t_i|E, W, a_{<i}, t_{<i})$ is estimated at the step/word i for trigger and argument predictions (i.e, in the last two components *trigger classifier* and *argument role classifier* of the overall architecture). We describe how those probabilities are computed using the hidden vectors h_i and the word embeddings d_i in the following. Note that the modeling of $P(a_i, t_i|E, W, a_{<i}, t_{<i})$ enables the use of the information from $a_{<i}$ and $t_{<i}$ to reveal the inter-dependencies among the multiple events appearing in the input sentence to better predict a_i and t_i.

Throughout this paper, we will refer to "*the local context D_i of a token w_i in W*" as the concatenation vector of the word embeddings of the words in a window u of w_i in W, i.e: $D_i = [d_{i-u}, \ldots, d_i, \ldots, d_{i+u}]$ where zero vectors are padded if the index is out of range.

Entity Mention Detector   For entity mention prediction, the probability $P(E|W)$ can be decomposed into:

$$P(E|W) = P(e_1|W)P(e_2|W, e_{<2}) \ldots P(e_n|W, e_{<n})$$

where $e_{<t} = e_1, e_2, \ldots, e_{i-1}$.

In this work, for each word w_i, we estimate $P(e_i|W, e_{<i})) = FF^{EMD}(R_i^{EMD})$ where FF^{EMD} is a feed-forward neural network followed by a softmax layer to transform the feature representation R_i^{EMD} for w_i in EMD into a probability distribution over the possible entity type labels for w_i. The feature representation R_i^{EMD} is, in turn, computed by concatenating the hidden vector h_i and the local context D_i for w_i: $R_i^{EMD} = [h_i, D_i]$.

Note that in the R_i^{EMD} representation for w_i, we do not use any information about the entity type prediction made for w_{i-1} as we find it not effective for our joint model from the development experiments. However, this might cause the orphan label issue (i.e, the I (inside) label for an entity type is not preceded by the B (beginning) label of the corresponding entity type). We prevent this issue by generating a transition score matrix between the entity type labels that penalizes any transitions to an I label but not from the corresponding B label. The Viterbi decoding algorithm is then employed to find the best predicted entity label sequence $E^P = e_1^p, e_2^p, \ldots, e_n^p$ for W based on the scores $P(e_i|W, e_{<i}))$ $(1 \leq i \leq n)$ and the generated transition matrix (Ma and Hovy 2016; He et al. 2017).

Trigger and Argument Prediction   Once the entity type label for every word in W has been decided, we continue with the predictions of event triggers and arguments in the *trigger classifier* and *argument role classifier* components. As mentioned above, this step is done sequentially over the sentence from left to right. At the current word/step i, we attempt to compute the probability $P(a_i, t_i|E, W, a_{<i}, t_{<i})$ that is decomposed into:

$$
\begin{aligned}
P(a_i, t_i|E, W, a_{<i}, t_{<i}) = {} & P(t_i|E, W, a_{<i}, t_{<i}) \\
& \times P(a_{i1}|E, W, a_{<i}, t_{<i+1}) \\
& \times P(a_{i2}|E, W, a_{i,<2}, a_{<i}, t_{<i+1}) \\
& \ldots \\
& \times P(a_{in}|E, W, a_{i,<n}, a_{<i}, t_{<i+1})
\end{aligned}
$$

where $a_{i,<j} = a_{i,1}, a_{i,2}, \ldots, a_{i,j-1}$.

In this product, the term $P(t_i|E, W, a_{<i}, t_{<i})$ is to predict the event type that the current word w_i is triggering. Note that this can output the "*Other*" type to indicate that the current word is not an event trigger. The term $P(a_{ij}|E, W, a_{i,<j}, a_{<i}, t_{<i+1})$, on the other hand, predict the role that the entity mention with the beginning token of w_j plays in the event mention associated with w_i (i.e, the current event mention). Note that $P(a_{ij}|E, W, a_{i,<j}, a_{<i}, t_{<i+1})$ is only meaningful if w_i is a trigger word and w_j is the beginning token of some entity mention in the sentence. In other cases, we can simply skip the computation for $P(a_{ij}|E, W, a_{i,<j}, a_{<i}, t_{<i+1})$. During the training phase, we use the golden entity mentions to decide which tokens start an entity mention while in the evaluation phase, the predicted entity type labels $E^P = e_1^p, e_2^p, \ldots, e_n^p$ in the previous step is used for this purpose.

In order to produce $P(t_i|E, W, a_{<i}, t_{<i})$, we compute the feature representation R_i^{ED} for the current word w_i feed it into a feed-forward neural network with softmax FF^{ED}, resulting in a probability distributions over the possible event types: $P(t_i|E, W, a_{<i}, t_{<i}) = FF^{ED}(R_i^{ED})$. Similar to EMD, we also compute the representation R_i^{ED} by: $R_i^{ED} = [h_i, D_i]$.

A greedy decoder is applied to decide the predicted event type t_i^p for the current word: $t_i^p = \text{argmax } P(t_i|E, W, a_{<i}, t_{<i})$.

Regarding the argument role distribution $P(a_{ij}|E, W, a_{i,<j}, a_{<i}, t_{<i+1})$, we also compute it with the feed-forward network FF^{ARP} and the feature representation R_{ij}^{ARP}: $P(a_{ij}|E, W, a_{i,<j}, a_{<i}, t_{<i+1}) = FF^{ARP}(R_{ij}^{ARP})$. However, $R_{i,j}^{ARP}$ in this case is computed as:

$$R_{ij}^{ARP} = [h_i, D_i, h_j, D_j, V(e_i^p), V(t_j^p), M_i, B_{ij}] \qquad (1)$$

where $V(x)$ is the function that converts a label x into an one-hot vector to represent the label. Note that during the training process, e_i^p and t_i^p would be set to the golden labels from the training data. M_i is the binary vector to indicate the event types and argument roles that appear before step i in W. Finally, B_{ij} is the binary vector inherited from (Li, Ji, and Huang 2013; Nguyen, Cho, and Grishman 2016a) to capture the discrete structures/features for argument prediction between the tokens i and j in the sentence (i.e, the shortest dependency paths, the context words etc.). Note that different from the prior work (Li, Ji, and Huang 2013; Nguyen, Cho, and Grishman 2016a), B_{ij} does not contain any features related to the entity types or subtypes of the entity mentions as these features are not available to us at the beginning. We instead resort to the predicted entity type e_i^p as demonstrated in Equation 1. We also apply the greedy strategy to predict the argument role a_{ij}^p in this case: $a_{ij}^p = \text{argmax } P(a_{ij}|E, W, a_{i,<j}, a_{<i}, t_{<i+1})$. This completes the description of our joint model for EE.

Training

We train the joint model by optimizing the negative log-likelihood function $C(A, T, E, W) =$

$-\log P(A, T, E|W)$. In order to encourage the loss terms for EMD, ED and ARP to converge at the same time, we penalize these terms differently in $C(A, T, E, W)$, leading to the following loss function in this work:

$$
C^*(A, T, E, W) =
$$
$$
-\alpha \sum_{i=1}^{n} \log P(e_i|W, e_{<i})
$$
$$
-\beta \sum_{i=1}^{n} \log P(t_i|E, W, a_{<i}, t_{<i})
$$
$$
-\gamma \sum_{i=1}^{n} \sum_{j=1}^{n} \log P(a_{ij}|E, W, a_{i,<j}, a_{<i}, t_{<i+1})
$$

where α, β and γ are the hyper-parameters. We use the SGD algorithm to optimize the parameters with mini-batches and the *Adadelta* update rules (Zeiler 2012). The gradients are computed with back-propagation while the parameters are rescaled if their Frobenius norms exceed a hyper-parameter.

Experiments

Dataset, Parameters, and Resources

We evaluate the proposed model on the ACE 2005 dataset. In order to ensure a fair comparison, we use the same data split with the prior work on this dataset (Li, Ji, and Huang 2013; Nguyen, Cho, and Grishman 2016a; Nguyen et al. 2016b; Yang and Mitchell 2016; Sha et al. 2018) in which 40 newswire documents are used for the test set, 30 other documents are reserved for the development set, and the remaining 529 documents form the training set. We utilize the Stanford CoreNLP to do the pre-processing for the sentences (i.e, POS tagging, chunking and dependency parsing). The pre-trained word embeddings are obtained from (Nguyen, Cho, and Grishman 2016a).

Regarding the hyper-parameters, the word embeddings have the dimension of 300; the number of hidden units in the encoding RNNs is 300; and the window for local context u is 2. We use the feed-forward neural networks with one hidden layer of 600 hidden units for FF^{EMD}, FF^{ED} and FF^{ARP}. The mini-batch size is 50 while the Frobenius norm for the parameters norms is 3. These values give us best the results on the development set. For the penalty coefficients in the objective function, the best values we obtained from the development data is $\alpha = 0.5, \beta = 1.0, \gamma = 0.5$. We also implement dropouts on the input word embeddings and the hidden vectors of the feed-forward networks with a rate of 0.5 (tuned on the development set). Finally, the same correctness criteria with the previous work (Nguyen, Cho, and Grishman 2016a; Yang and Mitchell 2016; Sha et al. 2018) is applied when we evaluate the predicted results.

Comparing to the State of the Art for Trigger and Argument Predictions

In order to evaluate the effectiveness of the proposed model for event trigger and argument predictions, we compare the proposed model (called *Joint3EE*) with the following baselines:

1. *StagedMaxent*: This is a feature-based pipelined baseline presented in (Yang and Mitchell 2016) (i.e, performing the three subtasks separately). The EMD subtask is solved by a CRF tagger. All the three subtasks are based on feature engineering.

2. *Pipelined-Feature*: This is the feature-based EE system that uses the same CRF tagger as *StagedMaxent*'s to annotate entity mentions. The results are passed to the joint model for ED and ARP in (Li, Ji, and Huang 2013). Similar to the CRF tagger, this joint model also employs complicated feature engineering. This baseline is reported in (Yang and Mitchell 2016) and functions as *the state-of-the-art pipelined model for EE using feature engineering*.

3. *Pipelined-Deep-Learning*: This baseline also first extracts entity mentions and then uses the outputs in a joint model for ED and ARP (pipelined). However, the EMD model and the joint model for ED and ARP are based on deep learning in this case. In particular, the EMD model for this baseline is inherited from the EMD component of this work while the joint model for ED and ARP is provided by (Nguyen, Cho, and Grishman 2016a). The performance of the EMD model when it is trained independently for EMD is shown in Table 2 (i.e, *EMD-Pipelined-DL*).

4. *Joint-Feature-Sentence*: This is the joint inference system that models the three subtasks of EE in a single model proposed in (Yang and Mitchell 2016). This model only considers the structures within a single event.

5. *Joint-Feature-Document*: This system is similar to *Joint-Feature-Sentence* except that it goes beyond the sentence level and exploits the event-event dependencies at the document level (Yang and Mitchell 2016). This is *the state-of-the-art joint model for EE using feature engineering*.

6. *NP-Candidate-Deep-Learning* (Sha et al. 2018): This method uses the existing tools to extract noun phrases and treat them as the argument candidates for the events. It is then followed by a dependency bridge recurrent neural network with argument interaction modeling to jointly perform ED and ARP for event extraction. This method currently has *the best reported performance on ED and ARP for EE among the methods that do not assume golden entity mentions with the ACE 2005 dataset*. The EMD task is not considered in this method.

Table 1 reports the performance of the systems in terms of precisions (P), recalls (R) and F1 scores (F). The first observation is that the performance of the joint deep learning model for ED and ARP in (Nguyen, Cho, and Grishman 2016a) with predicted entity mentions (i.e, *Pipelined-Deep-Learning* with 67.0% and 37.4% for trigger and argument role classification respectively) is much worse than that with perfect entity mentions in (Nguyen, Cho, and Grishman 2016a) (i.e, 69.3% and 55.4% for trigger and argument role classification respectively). This is consistent with the significant performance drop of the joint feature-based model for ED and ARP (reported in (Li, Ji, and Huang 2013)) when entity mentions are predicted. Such pieces of evidence along with the significantly better performance of the fully joint models for EE (i.e, *Joint-*

Model	Event Trigger Identification			Event Trigger Classification			Event Argument Identification			Argument Role Classification		
	P	R	F	P	R	F	P	R	F	P	R	F
StagedMaxent	73.9	66.5	70.0	70.4	63.3	66.7	75.7	20.2	31.9	71.2	19.0	30.0
Pipelined-Feature	76.6	58.7	66.5	74.0	56.7	64.2	74.6	25.5	38.0	68.8	23.5	35.0
Pipelined-Deep-Learning	72.7	65.9	69.1	70.4	63.9	67.0	61.7	42.1	50.1	46.0	31.4	37.4
Joint-Feature-Sentence	76.9	63.8	69.7	74.7	62.0	67.7	72.4	37.2	49.2	69.9	35.9	47.4
Joint-Feature-Document†	77.6	65.4	71.0	75.1	63.3	68.7	73.7	38.5	50.6	70.6	36.9	48.4
NP-Candidate-Deep-Learning	-	-	-	-	-	69.6	-	-	57.2	-	-	50.1
Joint3EE	70.5	74.5	**72.5**	68.0	71.8	**69.8**	59.9	59.8	**59.9**	52.1	52.1	**52.1**

Table 1: Performance on the ACE 2005 test set. The comparison between *Joint3EE* and *Pipelined-Deep-Learning* is significant with $p < 0.05$. "†" designates the systems with document level information.

Feature-Document and *Joint3EE*) over the pipelined models (i.e, *StagedMaxEnt*, *Pipelined-Feature* and *Pipelined-Deep-Learning*) in Table 1 demonstrate the need to jointly perform EMD with ED and ARP to improve the EE performance. We also see that *Pipelined-Deep-Learning* outperforms *Pipelined-Feature* and *Joint3EE* is significantly better than *Joint-Feature-Document* with respect to all the F1 scores in Table 1. These facts testify to the benefits of deep learning over the feature-based models for EE no matter which approach we take (i.e, pipelined or joint inference). Finally, comparing *Joint3EE* and the current state-of-the-art model *NP-Candidate-Deep-Learning*, we see that *Joint3EE* is superior to *NP-Candidate-Deep-Learning* on both trigger and argument prediction. The improvement is significant on event argument identification and argument role classification (an improvement of 2.7% for event argument identification and 2.0% for argument role classification on the absolute F1 scores), clearly demonstrating the effectiveness of the proposed deep learning method to jointly model the three subtasks for EE.

Performance of EMD

This section evaluates the EMD performance of the proposed joint model (called *EMD-Joint3EE*). The following baselines are chosen for comparison:

1. *EMD-CRF*: This is the performance of the CRF tagger for EMD used in the pipelined models *StagedMaxent* and *Pipelined-Feature*. It is implemented in (Yang and Mitchell 2016).

2. *EMD-Pipelined-DL*: This is the performance of the deep learning EMD module used in *Pipelined-Deep-Learning* that resembles the EMD component of the proposed model *Joint3EE*, but is trained separately from ED and ARP.

3. *EMD-Joint-Feature*: This corresponds to the EMD module in *Joint-Feature-Document* (Yang and Mitchell 2016) that is trained jointly with ED and ARP in a single model based on feature engineering. It is currently the state-of-the-art EMD performance in the setting for EE.

Table 2 shows the performance of the models. First, we can see from the table that the performance of the proposed model *EMD-Joint3EE* is better than that of *EMD-Pipelined-DL*. The performance improvement is 0.8% on the abso-

Model	P	R	F
EMD-CRF	85.5	73.5	79.1
EMD-Pipelined-DL	80.6	80.3	80.4
EMD-Joint-Feature	82.4	79.2	80.7
EMD-Joint3EE	82.0	80.4	**81.2**

Table 2: Entity Mention Detection Performance

lute F1 score and significant with $p < 0.05$. Second, we also see that *EMD-Joint3EE* outperforms the current state-of-the-art joint model *EMD-Joint-Feature* with an improvement of 0.5% on the F1 score. Such evidence confirms the benefits of jointly modeling EMD with ED and ARP via deep learning to improve the overall performance.

An intriguing observation is that the performance difference for EMD between *Pipelined-Deep-Learning* and *Joint3EE* is moderate (i.e, 0.8% in Table 2) while that difference for ARP is substantial (i.e, 13.9% in Table 1). Among several reasons, *Pipelined-Deep-Learning* employs the joint model for ED and ARP in (Nguyen, Cho, and Grishman 2016a) that relies on the discrete features only available to the manually annotated entity mentions such as the entity subtypes for the entity mentions (i.e, "*Crime*", "*Job-Title*" and "*Numeric*" for values). As demonstrated in Table 3, such information is very helpful for ARP. However, it is not available in our setting of predicted entity mentions (i.e, only the boundaries and the entity types are predicted), causing the poor ARP performance of *Pipelined-Deep-Learning*.

Model	P	R	F
(Nguyen, Cho, and Grishman 2016a)	54.2	56.7	55.4
Without Entity Subtype	46.2	48.8	47.5

Table 3: Performance on argument role classification of the joint model in (Nguyen, Cho, and Grishman 2016a) (using perfect entity mentions). The absence of the entity subtype information reduces the F1 score of the joint model in (Nguyen, Cho, and Grishman 2016a) by 7.9%.

For the *Joint3EE* model, although such discrete fine-grained information also does not exist explicitly, the shared

hidden vectors h_i across subtasks can learn to encode that information implicitly, thereby compensating the lack of information and improving the ARP performance.

The Effect of External Features

There are two main sources of external features employed in *Joint3EE*, i.e, the binary vectors for POS, chunking and dependency parsing information in the sentence encoding, and the binary features B_{ij} for argument role prediction in Equation 1. This section evaluates the effect of such features on the model performance to see how far we can reach an end-to-end model for joint EE with deep learning. Table 4 presents the performance of the proposed model when such features are excluded from the model (i.e, resulting in an end-to-end model for EE (called "*End-to-end-DL*") that does not use any external hand-designed features from the NLP toolkits. We also include the performance of the state-of-the-art models (i.e, *Joint-Feature-Document* (Yang and Mitchell 2016) and *NP-Candidate-Deep-Learning* (Sha et al. 2018)) that employ predicted entity mentions for EE to facilitate the comparison.

Feature	Entity	Trigger	Argument
Joint3EE	81.2	69.8	52.1
End-to-end-DL	79.5	68.7	50.3
Joint-Feature-Document	80.7	68.7	48.4
NP-Candidate-Deep-Learning	-	69.6	50.1

Table 4: F1 scores on classification for entity mentions (EMD), event triggers (ED) and arguments (ARP).

The first observation is that the external features are useful for the joint model *Joint3EE* as eliminating such features downgrades the performance over all the subtasks EMD, ED and ARP (i.e. comparing *joint3E* and *End-to-end-DL*). However, the performance reduction due to this feature removal is not dramatic and the performance of the *End-to-end-DL* system is still comparable with that of the current state-of-the-art models *Joint-Feature-Document* and *NP-Candidate-Deep-Learning*. In particular, *End-to-end-DL* is only 1.2% worse than *Joint-Feature-Document* on EMD and 0.9% worse than *NP-Candidate-Deep-Learning* on ED. Regarding ARP, *End-to-end-DL* even significantly outperforms *Joint-Feature-Document* with 1.9% performance improvement. These are remarkable facts given that *End-to-end-DL* does not use any external and manually-generated features while *Joint-Feature-Document* and *NP-Candidate-Deep-Learning* extensively rely on such external features to perform well (e.g, dependency parsing, NP chunking, gazetteers etc.). We consider this as a strong promise toward a state-of-the-art end-to-end system for EE for which the joint model in this work can be used as a good starting point.

Error Analysis

In order to analyze the operation of *Joint3EE* with respect to ED, we notice from Table 1 that the trigger classification performance (i.e, 69.8%) is quite close the trigger identification

MISSED		INCORRECT	
Label	Percent	Label	Percent
Attack	16.1%	End-Position	18.2%
Transfer-Ownership	12.5%	Attack	17.5%
Transport	12.5%	Transport	17.5%
Total	41.1%	Total	53.2%

Table 5: Top three event types for trigger errors.

performance (i.e, 72.5%). This suggests that the main source of errors for event triggers comes from the failure to identify the trigger words. To this end, we examine the outputs of *Joint3EE* on the test set to determine the contributions of each event type to the trigger identification errors. Two types of errors arise in this case: (i) missing an event trigger in the test set (called *MISSED*), and (ii) incorrectly detecting an event trigger (called *INCORRECT*). Table 5 shows the top three event types appearing in these two types of errors and their corresponding percents over the total numbers of errors. These top three event types account for 41.1% of the *MISSED* errors and 53.2% of the *INCORRECT* errors. *Attack* and *Transport* are the types that are present frequently in both types of errors. A closer look at the errors reveals that the "*MISSED*" errors mostly correspond to the trigger words not appearing in the training data, such as the word "*intifada*" (of type *Attack*) in the following sentence:

 . . . *had freedom of movement with cars and weapons since the start of the* **intifada**" . . .

The *INCORRECT* errors, on the other hand, belong to the confusable context that requires better modeling of the context. For instance, the word "*fire*" in the following sentence can be easily misinterpreted as an *Attack* event trigger by the models (due to its context with the word "*car*"):

 . . . *also take over GE's US car and* **fire** *insurance operations, the reports said.*

Regarding the argument prediction, we find that a large number of arguments (i.e, 209 arguments) are identified correctly, but cannot be classified properly by *Joint3EE* (i.e, Table 1). Among those 209 arguments, there are 50 cases (23.9%) that *Joint3EE* detects with the correct argument role, but assigns incorrect event types. The remaining 159 arguments (76.1%) are associated with incorrect roles for which only 24 arguments (15.1%) also have incorrect entity types. Consequently, the major problem for the incorrect argument classification is due to the confusion of the model on the different roles of arguments. The most frequent role confusions are between *Place* vs *Destination*, *Origin* vs *Destination*, and *Seller* vs *Buyer*. The distinction between these pairs of roles would also require better mechanisms/network architectures to model the input context.

Conclusion

We present a novel deep learning method for EE. Our model features the joint modeling of EMD, ED and ARP with the shared hidden representations across the three subtasks to

enable the communications among them. We achieve the state-of-the-art performance for EE with predicted entity mentions. In the future, we plan to improve the end-to-end model so EE can be solved from just the raw sentences and the word embeddings.

References

Ahn, D. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*.

Chen, Y.; Xu, L.; Liu, K.; Zeng, D.; and Zhao, J. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL-IJCNLP*.

Chen, Y.; Liu, S.; Zhang, X.; Liu, K.; and Zhao, J. 2017. Automatically labeled data generation for large scale event extraction. In *ACL*.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*.

Grishman, R.; Westbrook, D.; and Meyers, A. 2005. Nyu's english ace 2005 system description. In *ACE 2005 Evaluation Workshop*.

Gupta, P., and Ji, H. 2009. Predicting unknown time arguments based on cross-event propagation. In *ACL-IJCNLP*.

He, L.; Lee, K.; Lewis, M.; and Zettlemoyer, L. 2017. Deep semantic role labeling: What works and what's next. In *ACL*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. In *Neural Computation*.

Hong, Y.; Zhang, J.; Ma, B.; Yao, J.; Zhou, G.; and Zhu, Q. 2011. Using cross-entity inference to improve event extraction. In *ACL*.

Huang, R., and Riloff, E. 2012. Modeling textual cohesion for event extraction. In *AAAI*.

Huang, L.; Ji, H.; Cho, K.; and Voss, C. R. 2018. Zero-shot transfer learning for event extraction. In *arXiv preprint arXiv:1707.01066*.

Ji, H., and Grishman, R. 2008. Refining event extraction through cross-document inference. In *ACL*.

Judea, A., and Strube, M. 2016. Incremental global event extraction. In *COLING*.

Li, Q.; Ji, H.; Hong, Y.; and Li, S. 2014b. Constructing information networks using one single model. In *EMNLP*.

Li, X.; Nguyen, T. H.; Cao, K.; and Grishman, R. 2015. Improving event detection with abstract meaning representation. In *Proceedings of ACL-IJCNLP Workshop on Computing News Storylines (CNewS)*.

Li, Q.; Ji, H.; and Huang, L. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Liao, S., and Grishman, R. 2010. Using document level cross-event inference to improve event extraction. In *ACL*.

Liao, S., and Grishman, R. 2011. Acquiring topic features to improve event extraction: in pre-selected and balanced collections. In *RANLP*.

Liu, S.; Chen, Y.; Liu, K.; and Zhao, J. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *ACL*.

Liu, J.; Chen, Y.; Liu, K.; and Zhao, J. 2018. Event detection via gated multilingual attention mechanism. In *AAAI*.

Lu, W., and Nguyen, T. H. 2018. Similar but not the same: Word sense disambiguation improves event detection via neural representation matching. In *EMNLP*.

Ma, X., and Hovy, E. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*.

McClosky, D.; Surdeanu, M.; and Manning, C. 2011. Event extraction as dependency parsing. In *BioNLP Shared Task Workshop*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. In *ICLR*.

Miwa, M.; Thompson, P.; Korkontzelos, I.; and Ananiadou, S. 2014. Comparable study of event extraction in newswire and biomedical domains. In *COLING*.

Nguyen, T. H., and Grishman, R. 2015b. Event detection and domain adaptation with convolutional neural networks. In *ACL-IJCNLP*.

Nguyen, T. H., and Grishman, R. 2016d. Modeling skip-grams for event detection with convolutional neural networks. In *EMNLP*.

Nguyen, T. H., and Grishman, R. 2018a. Graph convolutional networks with argument-aware pooling for event detection. In *AAAI*.

Nguyen, T. H.; Fu, L.; Cho, K.; and Grishman, R. 2016b. A two-stage approach for extending event detection to new types via neural networks. In *Proceedings of the 1st ACL Workshop on Representation Learning for NLP (RepL4NLP)*.

Nguyen, T. H.; Cho, K.; and Grishman, R. 2016a. Joint event extraction via recurrent neural networks. In *NAACL*.

Patwardhan, S., and Riloff, E. 2009. A unified model of phrasal and sentential evidence for information extraction. In *EMNLP*.

Poon, H., and Vanderwende, L. 2010. Joint inference for knowledge extraction from biomedical literature. In *NAACL-HLT*.

Riedel, S., and McCallum, A. 2011a. Fast and robust joint models for biomedical event extraction. In *EMNLP*.

Riedel, S., and McCallum, A. 2011b. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *BioNLP Shared Task 2011 Workshop*.

Riedel, S.; Chun, H.-W.; Takagi, T.; and Tsujii, J. 2009. A markov logic approach to bio-molecular event extraction. In *BioNLP 2009 Workshop*.

Sha, L.; Qian, F.; Chang, B.; and Sui, Z. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *AAAI*.

Venugopal, D.; Chen, C.; Gogate, V.; and Ng, V. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *EMNLP*.

Yang, B., and Mitchell, T. M. 2016. Joint extraction of events and entities within a document context. In *NAACL-HLT*.

Zeiler, M. D. 2012. Adadelta: An adaptive learning rate method. In *CoRR, abs/1212.5701*.