Joint Event Extraction via Structured Prediction with Global Features

Qi Li **Heng Ji** **Liang Huang**
Departments of Computer Science and Linguistics
The Graduate Center and Queens College
City University of New York
New York, NY 10016, USA
{liqiearth, hengjicuny, liang.huang.sh}@gmail.com

Abstract

Traditional approaches to the task of ACE event extraction usually rely on sequential pipelines with multiple stages, which suffer from error propagation since event triggers and arguments are predicted in isolation by independent local classifiers. By contrast, we propose a joint framework based on structured prediction which extracts triggers and arguments together so that the local predictions can be mutually improved. In addition, we propose to incorporate global features which explicitly capture the dependencies of multiple triggers and arguments. Experimental results show that our joint approach with local features outperforms the pipelined baseline, and adding global features further improves the performance significantly. Our approach advances state-of-the-art sentence-level event extraction, and even outperforms previous argument labeling methods which use external knowledge from other sentences and documents.

1 Introduction

Event extraction is an important and challenging task in Information Extraction (IE), which aims to discover event triggers with specific types and their arguments. Most state-of-the-art approaches (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011) use sequential pipelines as building blocks, which break down the whole task into separate subtasks, such as trigger identification/classification and argument identification/classification. As a common drawback of the staged architecture, errors in upstream component are often compounded and propagated to the downstream classifiers. The downstream components, however, cannot impact earlier decisions. For example, consider the following sentences with an ambiguous word "fired":

(1) In Baghdad, a cameraman **died** when an *American tank* **fired** on the Palestine Hotel.

(2) He has **fired** his *air defense chief*.

In sentence (1), "fired" is a trigger of type *Attack*. Because of the ambiguity, a local classifier may miss it or mislabel it as a trigger of *End-Position*. However, knowing that "tank" is very likely to be an *Instrument* argument of *Attack* events, the correct event subtype assignment of "fired" is obviously *Attack*. Likewise, in sentence (2), "air defense chief" is a job title, hence the argument classifier is likely to label it as an *Entity* argument for *End-Position* trigger.

In addition, the local classifiers are incapable of capturing inter-dependencies among multiple event triggers and arguments. Consider sentence (1) again. Figure 1 depicts the corresponding event triggers and arguments. The dependency between "fired" and "died" cannot be captured by the local classifiers, which may fail to attach "cameraman" to "fired" as a *Target* argument. By using global features, we can propagate the *Victim* argument of the *Die* event to the *Target* argument of the *Attack* event. As another example, knowing that an *Attack* event usually only has one *Attacker* argument, we could penalize assignments in which one trigger has more than one *Attacker*. Such global features cannot be easily exploited by a local classifier.

Therefore, we take a fresh look at this problem and formulate it, for the first time, as a *structured learning* problem. We propose a novel joint event extraction algorithm to predict the triggers and arguments simultaneously, and use the structured perceptron (Collins, 2002) to train the joint model. This way we can capture the dependencies between triggers and argument as well as explore

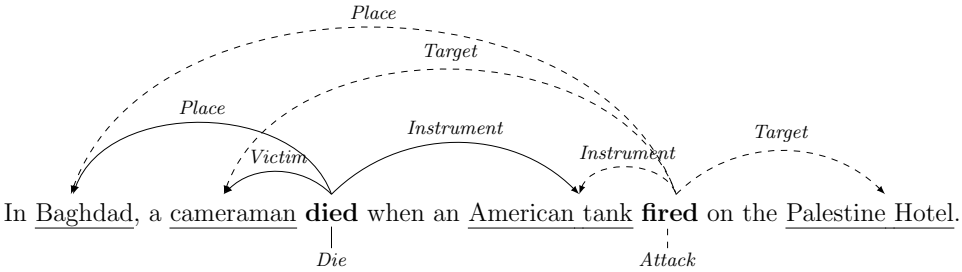

Figure 1: Event mentions of example (1). There are two event mentions that share three arguments, namely the *Die* event mention triggered by "died", and the *Attack* event mention triggered by "fired".

arbitrary global features over multiple local predictions. However, different from easier tasks such as part-of-speech tagging or noun phrase chunking where efficient dynamic programming decoding is feasible, here exact joint inference is intractable. Therefore we employ beam search in decoding, and train the model using the early-update perceptron variant tailored for beam search (Collins and Roark, 2004; Huang et al., 2012).

We make the following contributions:

1. Different from traditional pipeline approach, we present a novel framework for sentence-level event extraction, which predicts triggers and their arguments jointly (Section 3).

2. We develop a rich set of features for event extraction which yield promising performance even with the traditional pipeline (Section 3.4.1). In this paper we refer to them as local features.

3. We introduce various global features to exploit dependencies among multiple triggers and arguments (Section 3.4.2). Experiments show that our approach outperforms the pipelined approach with the same set of local features, and significantly advances the state-of-the-art with the addition of global features which brings a notable further improvement (Section 4).

2 Event Extraction Task

In this paper we focus on the event extraction task defined in Automatic Content Extraction (ACE) evaluation.[1] The task defines *8* event types and *33* subtypes such as *Attack*, *End-Position* etc. We introduce the terminology of the ACE event extraction that we used in this paper:

- Event mention: an occurrence of an event with a particular type and subtype.
- Event trigger: the word most clearly expresses the event mention.
- Event argument: an entity mention, temporal expression or value (e.g. *Job-Title*) that serves as a participant or attribute with a specific role in an event mention.
- Event mention: an instance that includes one event trigger and some arguments that appear within the same sentence.

Given an English text document, an event extraction system should predict event triggers with specific subtypes and their arguments from each sentence. Figure 1 depicts the event triggers and their arguments of sentence (1) in Section 1. The outcome of the entire sentence can be considered a graph in which each argument role is represented as a typed edge from a trigger to its argument.

In this work, we assume that argument candidates such as entities are part of the input to the event extraction, and can be from either gold standard or IE system output.

3 Joint Framework for Event Extraction

Based on the hypothesis that facts are interdependent, we propose to use structured perceptron with inexact search to jointly extract triggers and arguments that co-occur in the same sentence. In this section, we will describe the training and decoding algorithms for this model.

3.1 Structured perceptron with beam search

Structured perceptron is an extension to the standard linear perceptron for structured prediction, which was proposed in (Collins, 2002). Given a sentence instance $x \in \mathcal{X}$, which in our case is a sentence with argument candidates, the structured perceptron involves the following *decoding prob-*

[1] http://projects.ldc.upenn.edu/ace/

lem which finds the best configuration $z \in \mathcal{Y}$ according to the current model $\mathbf{w}$:

$$z = \underset{y' \in \mathcal{Y}(x)}{\mathbf{argmax}} \quad \mathbf{w} \cdot \mathbf{f}(x, y') \tag{1}$$

where $\mathbf{f}(x, y')$ represents the feature vector for instance x along with configuration y'.

The perceptron learns the model $\mathbf{w}$ in an online fashion. Let $\mathcal{D} = \{(x^{(j)}, y^{(j)})\}_{j=1}^{n}$ be the set of training instances (with j indexing the current training instance). In each iteration, the algorithm finds the best configuration z for x under the current model (Eq. 1). If z is incorrect, the weights are updated as follows:

$$\mathbf{w} = \mathbf{w} + \mathbf{f}(x, y) - \mathbf{f}(x, z) \tag{2}$$

The key step of the training and test is the decoding procedure, which aims to search for the best configuration under the current parameters. In simpler tasks such as part-of-speech tagging and noun phrase chunking, efficient dynamic programming algorithms can be employed to perform exact inference. Unfortunately, it is intractable to perform the exact search in our framework because: (1) by jointly modeling the trigger labeling and argument labeling, the search space becomes much more complex. (2) we propose to make use of arbitrary global features, which makes it infeasible to perform exact inference efficiently.

To address this problem, we apply beam-search along with early-update strategy to perform inexact decoding. Collins and Roark (2004) proposed the early-update idea, and Huang et al. (2012) later proved its convergence and formalized a general framework which includes it as a special case. Figure 2 describes the skeleton of perceptron training algorithm with beam search. In each step of the beam search, if the prefix of oracle assignment y falls out from the beam, then the top result in the beam is returned for early update. One could also use the standard-update for inference, however, with highly inexact search the standard-update generally does not work very well because of "invalid updates", i.e., updates that do not fix a violation (Huang et al., 2012). In Section 4.5 we will show that the standard perceptron introduces many invalid updates especially with smaller beam sizes, also observed by Huang et al. (2012).

To reduce overfitting, we used averaged parameters after training to decode test instances in our experiments. The resulting model is called averaged perceptron (Collins, 2002).

Input: Training set $\mathcal{D} = \{(x^{(j)}, y^{(j)})\}_{i=1}^{n}$, maximum iteration number T
Output: Model parameters $\mathbf{w}$

```
Initialization: Set w = 0;
for t ← 1...T do
    foreach (x, y) ∈ D do
        z ← beamSearch (x, y, w)
        if z ≠ y then
            w ← w + f(x, y_[1:|z|]) − f(x, z)
```

Figure 2: General perceptron training with beam-search (Huang et al., 2012). Here $y_{[1:i]}$ denotes the prefix of y that has length i, e.g., $y_{[1:3]} = (y_1, y_2, y_3)$.

3.2 Label sets

Here we introduce the label sets for trigger and argument in the model. We use $\mathcal{L} \cup \{\perp\}$ to denote the trigger label alphabet, where $\mathcal{L}$ represents the 33 event subtypes, and $\perp$ indicates that the token is not a trigger. Similarly, $\mathcal{R} \cup \{\perp\}$ denotes the argument label sets, where $\mathcal{R}$ is the set of possible argument roles, and $\perp$ means that the argument candidate is not an argument for the current trigger. It is worth to note that the set $\mathcal{R}$ of each particular event subtype is subject to the entity type constraints defined in the official ACE annotation guideline[2]. For example, the *Attacker* argument for an *Attack* event can only be one of *PER*, *ORG* and *GPE* (Geo-political Entity).

3.3 Decoding

Let $x = \langle (x_1, x_2, ..., x_s), \mathcal{E} \rangle$ denote the sentence instance, where x_i represents the i-th token in the sentence and $\mathcal{E} = \{e_k\}_{k=1}^{m}$ is the set of argument candidates. We use

$$y = (t_1,\, a_{1,1}, \ldots, a_{1,m},\, \ldots,\, t_s,\, a_{s,1}, \ldots, a_{s,m})$$

to denote the corresponding gold standard structure, where t_i represents the trigger assignment for the token x_i, and $a_{i,k}$ represents the argument role label for the edge between x_i and argument candidate e_k.

[2] http://projects.ldc.upenn.edu/ace/docs/English-Events-Guidelines_v5.4.3.pdf

$$y = (\underset{g(1)}{t_1},\ a_{1,1},\ a_{1,2},\ \underset{g(2)}{t_2},\ \underbrace{\underset{h(2,1)}{a_{2,1}},\ a_{2,2},}_{\text{arguments for } x_2}\ t_3,\ a_{3,1},\ \underset{h(3,2)}{a_{3,2}})$$

Figure 3: Example notation with $s = 3, m = 2$.

For simplicity, throughout this paper we use $y_{g(i)}$ and $y_{h(i,k)}$ to represent t_i and $a_{i,k}$, respectively. Figure 3 demonstrates the notation with $s = 3$ and $m = 2$. The variables for the toy sentence *"Jobs founded Apple"* are as follows:

$$x = \langle (\textit{Jobs}, \overbrace{\textit{founded}}^{x_2}, \textit{Apple}), \overbrace{\{\textit{Jobs}_{\text{PER}}, \textit{Apple}_{\text{ORG}}\}}^{\mathcal{E}} \rangle$$
$$y = (\perp, \perp, \perp, \underbrace{\textit{Start\_Org}}_{t_2}, \underbrace{\textit{Agent},\ \textit{Org}}_{\text{args for } \textit{founded}}, \perp, \perp, \perp)$$

Figure 4 describes the beam-search procedure with early-update for event extraction. During each step with token i, there are two sub-steps:

- **Trigger labeling** We enumerate all possible trigger labels for the current token. The linear model defined in Eq. (1) is used to score each partial configuration. Then the K-best partial configurations are selected to the beam, assuming the beam size is K.
- **Argument labeling** After the trigger labeling step, we traverse all configurations in the beam. Once a trigger label for x_i is found in the beam, the decoder searches through the argument candidates $\mathcal{E}$ to label the edges between each argument candidate and the trigger. After labeling each argument candidate, we again score each partial assignment and select the K-best results to the beam.

After the second step, the rank of different trigger assignments can be changed because of the argument edges. Likewise, the decision on later argument candidates may be affected by earlier argument assignments.

The overall time complexity for decoding is $O(K \cdot s \cdot m)$.

3.4 Features

In this framework, we define two types of features, namely local features and global features. We first introduce the definition of local and global features in this paper, and then describe the implementation details later. Recall that in the linear model defined in Eq. (1), $\mathbf{f}(x, y)$ denotes the features extracted from the input instance x along

Input: Instance $x = \langle (x_1, x_2, ..., x_s), \mathcal{E} \rangle$ and the oracle output y if for training.
K: Beam size.
$\mathcal{L} \cup \{\perp\}$: trigger label alphabet.
$\mathcal{R} \cup \{\perp\}$: argument label alphabet.
Output: 1-best prediction z for x

1 Set beam $\mathcal{B} \leftarrow [\epsilon]$ /*empty configuration*/
2 **for** $i \leftarrow 1...s$ **do**
3 $buf \leftarrow \{z' \circ l \mid z' \in \mathcal{B}, l \in \mathcal{L} \cup \{\perp\}\}$
$\mathcal{B} \leftarrow$K-best(buf)
4 **if** $y_{[1:g(i)]} \notin \mathcal{B}$ **then**
5 return $\mathcal{B}[0]$ /*for early-update*/
6 **for** $e_k \in \mathcal{E}$ **do** /*search for arguments*/
7 $buf \leftarrow \emptyset$
8 **for** $z' \in \mathcal{B}$ **do**
9 $buf \leftarrow buf \cup \{z' \circ \perp\}$
10 **if** $z'_{g(i)} \neq \perp$ **then** /*x_i is a trigger*/
11 $buf \leftarrow buf \cup \{z' \circ r \mid r \in \mathcal{R}\}$
12 $\mathcal{B} \leftarrow$K-best(buf)
13 **if** $y_{[1:h(i,k)]} \notin \mathcal{B}$ **then**
14 return $\mathcal{B}[0]$ /*for early-update*/
15 return $\mathcal{B}[0]$

Figure 4: Decoding algorithm for event extraction. $z \circ l$ means appending label l to the end of z. During test, lines 4-5 & 13-14 are omitted.

with configuration y. In general, each feature instance f in $\mathbf{f}$ is a function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, which maps x and y to a feature value. Local features are only related to predictions on individual trigger or argument. In the case of unigram tagging for trigger labeling, each local feature takes the form of $f(x, i, y_{g(i)})$, where i denotes the index of the current token, and $y_{g(i)}$ is its trigger label. In practice, it is convenient to define the local feature function as an indicator function, for example:

$$f_1(x, i, y_{g(i)}) = \begin{cases} 1 & \text{if } y_{g(i)} = \text{Attack and } x_i = \textit{"fire"} \\ 0 & \text{otherwise} \end{cases}$$

The global features, by contrast, involve longer range of the output structure. Formally, each global feature function takes the form of $f(x, i, k, y)$, where i and k denote the indices of the current token and argument candidate in decoding, respectively. The following indicator function is a simple example of global features:

$$f_{101}(x, i, k, y) = \begin{cases} 1 & \text{if } y_{g(i)} = \text{Attack and} \\ & y \text{ has only one } \textit{"Attacker"} \\ 0 & \text{otherwise} \end{cases}$$

Category	Type	Feature Description
Trigger	Lexical	*1.* unigrams/bigrams of the current and context words within the window of size 2 *2.* unigrams/bigrams of part-of-speech tags of the current and context words within the window of size 2 *3.* lemma and synonyms of the current token *4.* base form of the current token extracted from Nomlex (Macleod et al., 1998) *5.* Brown clusters that are learned from ACE English corpus (Brown et al., 1992; Miller et al., 2004; Sun et al., 2011). We used the clusters with prefixes of length 13, 16 and 20 for each token.
	Syntactic	*6.* dependent and governor words of the current token *7.* dependency types associated the current token *8.* whether the current token is a modifier of job title *9.* whether the current token is a non-referential pronoun
	Entity Information	*10.* unigrams/bigrams normalized by entity types *11.* dependency features normalized by entity types *12.* nearest entity type and string in the sentence/clause
Argument	Basic	*1.* context words of the entity mention *2.* trigger word and subtype *3.* entity type, subtype and entity role if it is a geo-political entity mention *4.* entity mention head, and head of any other name mention from co-reference chain *5.* lexical distance between the argument candidate and the trigger *6.* the relative position between the argument candidate and the trigger: {before, after, overlap, or separated by punctuation} *7.* whether it is the nearest argument candidate with the same type *8.* whether it is the only mention of the same entity type in the sentence
	Syntactic	*9.* dependency path between the argument candidate and the trigger *10.* path from the argument candidate and the trigger in constituent parse tree *11.* length of the path between the argument candidate and the trigger in dependency graph *12.* common root node and its depth of the argument candidate and parse tree *13.* whether the argument candidate and the trigger appear in the same clause

Table 1: Local features.

3.4.1 Local features

In general there are two kinds of local features:

Trigger features The local feature function for trigger labeling can be factorized as $f(x, i, y_{g(i)}) = p(x, i) \circ q(y_{g(i)})$, where $p(x, i)$ is a predicate about the input, which we call text feature, and $q(y_{g(i)})$ is a predicate on the trigger label. In practice, we define two versions of $q(y_{g(i)})$:

$$q_0(y_{g(i)}) = y_{g(i)} \text{ (event subtype)}$$
$$q_1(y_{g(i)}) = \text{event type of } y_{g(i)}$$

$q_1(y_{g(i)})$ is a backoff version of the standard unigram feature. Some text features for the same event type may share a certain distributional similarity regardless of the subtypes. For example, if the nearest entity mention is *"Company"*, the current token is likely to be *Personnel* no matter whether it is *End-Postion* or *Start-Position*.

Argument features Similarly, the local feature function for argument labeling can be represented as $f(x, i, k, y_{g(i)}, y_{h(i,k)}) = p(x, i, k) \circ q(y_{g(i)}, y_{h(i,k)})$, where $y_{h(i,k)}$ denotes the argument assignment for the edge between trigger word i and argument candidate e_k. We define two versions of $q(y_{g(i)}, y_{h(i,k)})$:

$$q_0(y_{g(i)}, y_{h(i,k)}) = \begin{cases} y_{h(i,k)} & \text{if } y_{h(i,k)} \text{ is } \textit{Place}, \\ & \textit{Time} \text{ or } \textit{None} \\ y_{g(i)} \circ y_{h(i,k)} & \text{otherwise} \end{cases}$$

$$q_1(y_{g(i)}, y_{h(i,k)}) = \begin{cases} 1 & \text{if } y_{h(i,k)} \neq \textit{None} \\ 0 & \text{otherwise} \end{cases}$$

It is notable that *Place* and *Time* arguments are applicable and behave similarly to all event subtypes. Therefore features for these arguments are not conjuncted with trigger labels. $q_1(y_{h(i,k)})$ can be considered as a backoff version of $q_0(y_{h(i,k))}$, which does not discriminate different argument roles but only focuses on argument identification. Table 1 summarizes the text features about the input for trigger and argument labeling. In our experiments, we used the Stanford parser (De Marneffe et al., 2006) to create dependency parses.

3.4.2 Global features

Table 2 summarizes the 8 types of global features we developed in this work. They can be roughly divided into the following two categories:

Category	Feature Description
Trigger	*1.* bigram of trigger types occur in the same sentence or the same clause *2.* binary feature indicating whether synonyms in the same sentence have the same trigger label *3.* context and dependency paths between two triggers conjuncted with their types
Argument	*4.* context and dependency features about two argument candidates which share the same role within the same event mention *5.* features about one argument candidate which plays as arguments in two event mentions in the same sentence *6.* features about two arguments of an event mention which are overlapping *7.* the number of arguments with each role type of an event mention conjuncted with the event subtype *8.* the pairs of time arguments within an event mention conjuncted with the event subtype

Table 2: Global features.

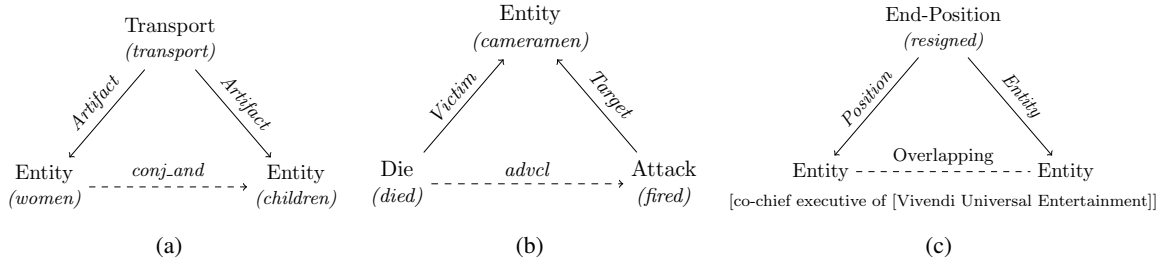

Figure 5: Illustration of global features (4-6) in Table 2.

Event	Probability
Attack	0.34
Die	0.14
Transport	0.08
Injure	0.04
Meet	0.02

Table 3: Top 5 event subtypes that co-occur with *Attack* event in the same sentence.

Trigger global feature This type of feature captures the dependencies between two triggers within the same sentence. For instance: feature (1) captures the co-occurrence of trigger types. This kind of feature is motivated by the fact that two event mentions in the same sentence tend to be semantically coherent. As an example, from Table 3 we can see that *Attack* event often co-occur with *Die* event in the same sentence, but rarely co-occur with *Start-Position* event. Feature (2) encourages synonyms or identical tokens to have the same label. Feature (3) exploits the lexical and syntactic relation between two triggers. A simple example is whether an *Attack* trigger and a *Die* trigger are linked by the dependency relation *conj_and*.

Argument global feature This type of feature is defined over multiple arguments for the same or different triggers. Consider the following sentence:

(3) Trains running to southern Sudan were used to ***transport*** abducted women and children.

The *Transport* event mention *"transport"* has two *Artifact* arguments, *"women"* and *"children"*. The dependency edge *conj_and* between *"women"* and *"children"* indicates that they should play the same role in the event mention. The triangle structure in Figure 5(a) is an example of feature (4) for the above example. This feature encourages entities that are linked by dependency relation *conj_and* to play the same role *Artifact* in any *Transport* event.

Similarly, Figure 5(b) depicts an example of feature (5) for sentence (1) in Section 1. In this example, an entity mention is *Victim* argument to *Die* event and *Target* argument to *Attack* event, and the two event triggers are connected by the typed dependency *advcl*. Here *advcl* means that the word *"fired"* is an adverbial clause modier of *"died"*.

Figure 5(c) shows an example of feature (6) for the following sentence:

(4) Barry Diller ***resigned*** as co-chief executive of Vivendi Universal Entertainment.

The job title *"co-chief executive of Vivendi Universal Entertainment"* overlaps with the *Organization* mention *"Vivendi Universal Entertainment"*. The feature in the triangle shape can be considered as a soft constraint such that if a *Job-Title* mention is a *Position* argument to an *End-Position* trigger, then the *Organization* mention

which appears at the end of it should be labeled as *Entity* argument for the same trigger.

Feature (7-8) are based on the statistics about different arguments for the same trigger. For instance, in many cases, a trigger can only have one *Place* argument. If a partial configuration mistakenly classifies more than one entity mention as *Place* arguments for the same trigger, then it will be penalized.

4 Experiments

4.1 Data set and evaluation metric

We utilized the ACE 2005 corpus as our testbed. For comparison, we used the same test set with *40* newswire articles (672 sentences) as in (Ji and Grishman, 2008; Liao and Grishman, 2010) for the experiments, and randomly selected *30* other documents (863 sentences) from different genres as the development set. The rest 529 documents (14, 840 sentences) are used for training.

Following previous work (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011), we use the following criteria to determine the correctness of an predicted event mention:

- A trigger is correct if its event subtype and offsets match those of a reference trigger.
- An argument is correctly *identified* if its event subtype and offsets match those of any of the reference argument mentions.
- An argument is correctly *identified* and *classified* if its event subtype, offsets and argument role match those of any of the reference argument mentions.

Finally we use *Precision (P)*, *Recall (R)* and *F-measure (F_1)* to evaluate the overall performance.

4.2 Baseline system

Chen and Ng (2012) have proven that performing identification and classification in one step is better than two steps. To compare our proposed method with the previous pipelined approaches, we implemented two Maximum Entropy (MaxEnt) classifiers for trigger labeling and argument labeling respectively. To make a fair comparison, the feature sets in the baseline are identical to the local text features we developed in our framework (see Figure 1).

4.3 Training curves

We use the *harmonic mean* of the trigger's F_1 measure and argument's F_1 measure to measure the performance on the development set.

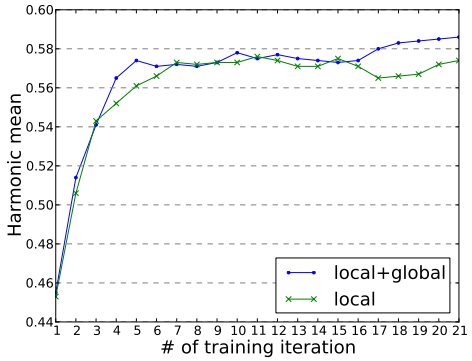

Figure 6: Training curves on dev set.

Figure 6 shows the training curves of the averaged perceptron with respect to the performance on the development set when the beam size is 4. As we can see both curves converge around iteration 20 and the global features improve the overall performance, compared to its counterpart with only local features. Therefore we set the number of iterations as 20 in the remaining experiments.

4.4 Impact of beam size

The beam size is an important hyper parameter in both training and test. Larger beam size will increase the computational cost while smaller beam size may reduce the performance. Table 4 shows the performance on the development set with several different beam sizes. When beam size = 4, the algorithm achieved the highest performance on the development set with trigger $\mathrm{F}_1 = 67.9$, argument $\mathrm{F}_1 = 51.5$, and harmonic mean $= 58.6$. When the size is increased to 32, the accuracy was not improved. Based on this observation, we chose beam size as 4 for the remaining experiments.

4.5 Early-update vs. standard-update

Huang et al. (2012) define "invalid update" to be an update that does not fix a violation (and instead reinforces the error), and show that it strongly (anti-)correlates with search quality and learning quality. Figure 7 depicts the percentage of invalid updates in standard-update with and without global features, respectively. With global features, there are numerous invalid updates when the

Beam size	1	2	4	8	16	32
Training time (sec)	993	2,034	3,982	8,036	15,878	33,026
Harmonic mean	57.6	57.7	**58.6**	58.0	57.8	57.8

Table 4: Comparison of training time and accuracy on the dev set.

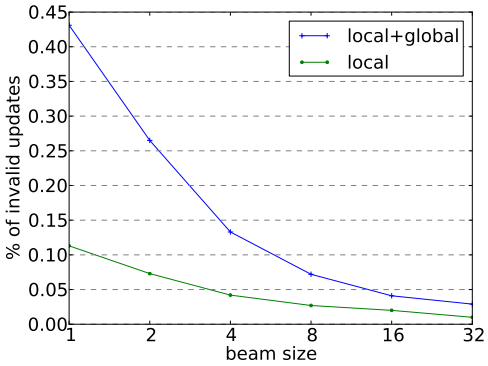

Figure 7: Percentage of the so-called "invalid updates" (Huang et al., 2012) in standard perceptron.

Strategy	F_1 on Dev		F_1 on Test	
	Trigger	Arg	Trigger	Arg
Standard ($b = 1$)	68.3	47.4	64.4	49.8
Early ($b = 1$)	**68.9**	**49.5**	**65.2**	**52.1**
Standard ($b = 4$)	**68.4**	50.5	67.1	51.4
Early ($b = 4$)	67.9	**51.5**	**67.5**	**52.7**

Table 5: Comparison between the performance (%) of standard-update and early-update with global features. Here b stands for beam size.

beam size is small. The ratio decreases monotonically as beam size increases. The model with only local features made much smaller numbers of invalid updates, which suggests that the use of global features makes the search problem much harder. This observation justify the application of early-update in this work. To further investigate the difference between early-update and standard-update, we tested the performance of both strategies, which is summarized in Table 5. As we can see the performance of standard-update is generally worse than early-update. When the beam size is increased ($b = 4$), the gap becomes smaller as the ratio of invalid updates is reduced.

4.6 Overall performance

Table 6 shows the overall performance on the blind test set. In addition to our baseline, we compare against the sentence-level system reported in Hong et al. (2011), which, to the best of our knowledge, is the best-reported system in the literature based on gold standard argument candidates. The proposed joint framework with local features achieves comparable performance for triggers and outperforms the staged baseline especially on arguments. By adding global features, the overall performance is further improved significantly. Compared to the staged baseline, it gains 1.6% improvement on trigger's F-measure and 8.8% improvement on argument's F-measure. Remarkably, compared to the cross-entity approach reported in (Hong et al., 2011), which attained 68.3% F_1 for triggers and 48.3% for arguments, our approach with global features achieves even better performance on argument labeling although we only used sentence-level information.

We also tested the performance with argument candidates automatically extracted by a high-performing name tagger (Li et al., 2012b) and an IE system (Grishman et al., 2005). The results are summarized in Table 7. The joint approach with global features significantly outperforms the baseline and the model with only local features. We also show that it outperforms the sentence-level baseline reported in (Ji and Grishman, 2008; Liao and Grishman, 2010), both of which attained 59.7% F_1 for triggers and 36.6% for arguments. Our approach aims to tackle the problem of sentence-level event extraction, thereby only used intra-sentential evidence. Nevertheless, the performance of our approach is still comparable with the best-reported methods based on cross-document and cross-event inference (Ji and Grishman, 2008; Liao and Grishman, 2010).

5 Related Work

Most recent studies about ACE event extraction rely on staged pipeline which consists of separate local classifiers for trigger labeling and argument labeling (Grishman et al., 2005; Ahn, 2006; Ji and Grishman, 2008; Chen and Ji, 2009; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2012a; Chen and Ng, 2012). To the best of our knowledge, our work is the first attempt to jointly model these two ACE event subtasks.

Methods	Trigger Identification (%)			Trigger Identification + classification (%)			Argument Identification (%)			Argument Role (%)		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Sentence-level in Hong et al. (2011)	N/A			67.6	53.5	59.7	46.5	37.15	41.3	41.0	32.8	36.5
Staged MaxEnt classifiers	76.2	60.5	67.4	74.5	59.1	65.9	74.1	37.4	49.7	65.4	33.1	43.9
Joint w/ local features	77.4	62.3	69.0	73.7	59.3	65.7	69.7	39.6	50.5	64.1	36.5	46.5
Joint w/ local + global features	76.9	65.0	**70.4**	73.7	62.3	**67.5**	69.8	47.9	**56.8**	64.7	44.4	**52.7**
Cross-entity in Hong et al. (2011)†	N/A			72.9	64.3	68.3	53.4	52.9	53.1	51.6	45.5	48.3

Table 6: Overall performance with gold-standard entities, timex, and values. †beyond sentence level.

Methods	Trigger F_1	Arg F_1
Ji and Grishman (2008) cross-doc Inference	67.3	42.6
Ji and Grishman (2008) sentence-level	59.7	36.6
MaxEnt classifiers	64.7 (↓1.2)	33.7 (↓10.2)
Joint w/ local	63.7 (↓2.0)	35.8 (↓10.7)
Joint w/ local + global	**65.6 (↓1.9)**	**41.8** (↓10.9)

Table 7: Overall performance (%) with predicted entities, timex, and values. ↓ indicates the performance drop from experiments with gold-standard argument candidates (see Table 6).

For the Message Understanding Conference (MUC) and FAS Program for Monitoring Emerging Diseases (ProMED) event extraction tasks, Patwardhan and Riloff (2009) proposed a probabilistic framework to extract event role fillers conditioned on the sentential event occurrence. Besides having different task definitions, the key difference from our approach is that their role filler recognizer and sentential event recognizer are trained independently but combined in the test stage. Our experiments, however, have demonstrated that it is more advantageous to do *both* training and testing with joint inference.

There has been some previous work on joint modeling for biomedical events (Riedel and McCallum, 2011a; Riedel et al., 2009; McClosky et al., 2011; Riedel and McCallum, 2011b). (McClosky et al., 2011) is most closely related to our approach. They casted the problem of biomedical event extraction as a dependency parsing problem. The key assumption that event structure can be considered as trees is incompatible with ACE event extraction. In addition, they used a separate classifier to predict the event triggers before applying the parser, while we extract the triggers and argument jointly. Finally, the features in the parser are edge-factorized. To exploit global features, they applied a MaxEnt-based global re-ranker. In comparison, our approach is a unified framework based on beam search, which allows us to exploit arbitrary global features efficiently.

6 Conclusions and Future Work

We presented a joint framework for ACE event extraction based on structured perceptron with inexact search. As opposed to traditional pipelined approaches, we re-defined the task as a structured prediction problem. The experiments proved that the perceptron with local features outperforms the staged baseline and the global features further improve the performance significantly, surpassing the current state-of-the-art by a large margin.

As shown in Table 7, the overall performance drops substantially when using predicted argument candidates. To improve the accuracy of end-to-end IE system, we plan to develop a complete joint framework to recognize entities together with event mentions for future work. Also we are interested in applying this framework to other IE tasks such as relation extraction.

Acknowledgments

This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053 (NS-CTA), U.S. NSF CAREER Award under Grant IIS-0953149, U.S. NSF EAGER Award under Grant No. IIS-1144111, U.S. DARPA Award No. FA8750-13-2-0041 in the "Deep Exploration and Filtering of Text" (DEFT) Program, a CUNY Junior Faculty Award, and Queens College equipment funds. 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

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Zheng Chen and Heng Ji. 2009. Language specific issue and feature exploration in chinese event extraction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 209–212.

Chen Chen and Vincent Ng. 2012. Joint modeling for chinese event extraction with rich linguistic features. In *COLING*, pages 529–544.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyu's english ace 2005 system description. In *Proceedings of ACE 2005 Evaluation Workshop*. Washington.

Yu Hong, Jianfeng Zhang, Bin Ma, Jian-Min Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of ACL*, pages 1127–1136.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL*, pages 254–262.

Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. 2012a. Employing compositional semantics and discourse consistency in chinese event extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1006–1016.

Qi Li, Haibo Li, Heng Ji, Wen Wang, Jing Zheng, and Fei Huang. 2012b. Joint bilingual name tagging for parallel corpora. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1727–1731.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of ACL*, pages 789–797.

Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. 1998. Nomlex: A lexicon of nominalizations. In *Proceedings of EURALEX*, volume 98, pages 187–193.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *Proceedings of ACL*, pages 1626–1635.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of HLT-NAACL*, volume 4, pages 337–342.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 151–160.

Sebastian Riedel and Andrew McCallum. 2011a. Fast and robust joint models for biomedical event extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–12.

Sebastian Riedel and Andrew McCallum. 2011b. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 46–50.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 41–49.

Ang Sun, Ralph Grishman, and Satoshi Sekine. 2011. Semi-supervised relation extraction with large-scale word clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 521–529.