Automatic Event Extraction with Structured Preference Modeling

Wei Lu and **Dan Roth**
University of Illinois at Urbana-Champaign
{luwei,danr}@illinois.edu

Abstract

This paper presents a novel sequence labeling model based on the latent-variable semi-Markov conditional random fields for jointly extracting argument roles of events from texts. The model takes in coarse mention and type information and predicts argument roles for a given event template.

This paper addresses the event extraction problem in a primarily unsupervised setting, where no labeled training instances are available. Our key contribution is a novel learning framework called *structured preference modeling* (PM), that allows arbitrary preference to be assigned to certain structures during the learning procedure. We establish and discuss connections between this framework and other existing works. We show empirically that the structured preferences are crucial to the success of our task. Our model, trained without annotated data and with a small number of structured preferences, yields performance competitive to some baseline supervised approaches.

1 Introduction

Automatic template-filling-based event extraction is an important and challenging task. Consider the following text span that describes an "Attack" event:

> *. . . North Korea's military may have fired a laser at a U.S. helicopter in March, a U.S. official said Tuesday, as the communist state ditched its last legal obligation to keep itself free of nuclear weapons . . .*

A partial event template for the "Attack" event is shown on the left of Figure 1. Each row shows an argument for the event, together with a set of its acceptable mention types, where the type specifies a high-level semantic class a mention belongs to.

The task is to automatically fill the template entries with texts extracted from the text span above. The correct filling of the template for this particular example is shown on the right of Figure 1.

Performing such a task without any knowledge about the semantics of the texts is hard. One typical assumption is that certain coarse mention-level information, such as mention boundaries and their semantic class (a.k.a. *types*), are available. E.g.:

> *. . .* [*North Korea's military*]$_{\text{ORG}}$ *may have fired* [*a laser*]$_{\text{WEA}}$ *at* [*a U.S. helicopter*]$_{\text{VEH}}$ *in* [*March*]$_{\text{TME}}$*, a U.S. official said Tuesday, as the communist state ditched its last legal obligation to keep itself free of nuclear weapons . . .*

Such mention type information as shown on the left of Figure 1 can be obtained from various sources such as dictionaries, gazetteers, rule-based systems (Strötgen and Gertz, 2010), statistically trained classifiers (Ratinov and Roth, 2009), or some web resources such as Wikipedia (Ratinov et al., 2011).

However, in practice, outputs from existing mention identification and typing systems can be far from ideal. Instead of obtaining the above ideal annotation, one might observe the following noisy and ambiguous annotation for the given event span:

> *. . .* [[*North Korea's*]$_{\text{GPE|LOC}}$ *military*]$_{\text{ORG}}$ *may have fired a laser at* [*a* [*U.S.*]$_{\text{GPE|LOC}}$ *helicopter*]$_{\text{VEH}}$ *in* [*March*]$_{\text{TME}}$*,* [*a* [*U.S.*]$_{\text{GPE|LOC}}$ *official*]$_{\text{PER}}$ *said* [*Tuesday*]$_{\text{TME}}$*, as* [*the communist state*]$_{\text{ORG|FAC|LOC}}$ *ditched its last legal obligation to keep* [*itself*]$_{\text{ORG}}$ *free of* [*nuclear weapons*]$_{\text{WEA}}$ *. . .*

Our task is to design a model to effectively select mentions in an event span and assign them with corresponding argument information, given such coarse

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 835–844,
Jeju, Republic of Korea, 8-14 July 2012.

Argument	Possible Types	Extracted Text
ATTACKER	GPE, ORG, PER	*N. Korea's military*
INSTRUMENT	VEH, WEA	*a laser*
PLACE	FAC, GPE, LOC	-
TARGET	FAC, GPE, LOC ORG, PER, VEH	*a U.S. helicopter*
TIME-WITHIN	TME	*March*

Figure 1: The partial event template for the Attack event (left), and the correct event template annotation for the example event span given in Sec 1 (right). We primarily follow the ACE standard in defining arguments and types.

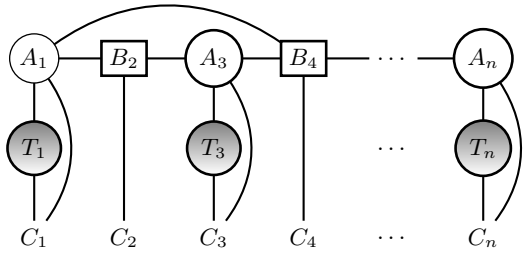

Figure 2: A simplified graphical illustration for the semi-Markov CRF, under a specific segmentation $S \equiv \overline{C_1 C_2 \dots C_n}$. In a supervised setting, only correct arguments are observed but their associated correct mention types are hidden (shaded).

and often noisy mention type annotations.

This work addresses this problem by making the following contributions:

- Naturally, we are interested in identifying the active mentions (the mentions that serve as arguments) and their correct boundaries from the data. This motivates us to build a novel latent-variable semi-Markov conditional random fields model (Sarawagi and Cohen, 2004) for such an event extraction task. The learned model takes in coarse information as produced by existing mention identification and typing modules, and jointly outputs selected mentions and their corresponding argument roles.
- We address the problem in a more realistic scenario where annotated training instances are not available. We propose a novel general learning framework called *structured preference modeling* (or *preference modeling*, PM), which encompasses both the fully supervised and the latent-variable conditional models as special cases. The framework allows arbitrary declarative structured preference knowledge to be introduced to guide the learning procedure in a primarily unsupervised setting.

We present our semi-Markov model and discuss our preference modeling framework in Section 2 and 3 respectively. We then discuss the model's relation with existing constraint-driven learning frameworks in Section 4. Finally, we demonstrate through experiments that structured preference information is crucial to model and present empirical results on a standard dataset in Section 5.

2 The Model

It is not hard to observe from the example presented in the previous section that dependencies between arguments can be important and need to be properly modeled. This motivates us to build a joint model for extracting the event structures from the text.

We show a simplified graphical representation of our model in Figure 2. In the graph, $C_1, C_2 \dots C_n$ refer to a particular segmentation of the event span, where C_1, C_3 ...correspond to mentions (e.g., "*North Korea's military*", "*a laser*") and C_2, C_4 ...correspond to in-between mention word sequences (we call them *gaps*) (e.g., "*may have fired*"). The symbols $T_1, T_3 \dots$ refer to mention types (e.g., GPE, ORG). The symbols $A_1, A_3 \dots$ refer to event arguments that carry specific roles (e.g., ATTACKER). We also introduce symbols $B_2, B_4 \dots$ to refer to inter-argument gaps. The event span is split into segments, where each segment is either linked to a mention type (T_i; these segments can be referred to as "argument segments"), or directly linked to an inter-argument gap (B_j; they can be referred to as "gap segments"). The two types of segments appear in the sequence in a strictly alternate manner, where the gaps can be of length zero. In the figure, for example, the segments C_1 and C_3 are identified as two argument segments (which are mentions of types T_1 and T_3 respectively) and are mapped to two "nodes", and the segment C_2 is identified as a gap segment that connects the two arguments A_1 and A_3. Note that no overlapping arguments are allowed in this model [1].

We use s to denote an event span and t to denote a specific realization (filling) of the event template. Templates consist of a set of arguments. Denote by h a particular mention boundary and type assignment for an event span, which gives us a specific segmentation of the given span. Following the conditional

[1] Extending the model to support certain argument overlapping is possible – we leave it for future work.

random fields model (Lafferty et al., 2001), we parameterize the conditional probability of the (t, h) pair given an event span s as follows:

$$P_\Theta(t, h|s) = \frac{e^{\mathbf{f}(s,h,t)\cdot\Theta}}{\sum_{t,h} e^{\mathbf{f}(s,h,t)\cdot\Theta}} \quad (1)$$

where $\mathbf{f}$ gives the feature functions defined on the tuple (s, h, t), and Θ defines the parameter vector.

Our objective function is the logarithm of the joint conditional probability of observing the template realization for the observed event span s:

$$\begin{aligned}\mathcal{L}(\Theta) &= \sum_i \log P_\Theta(t_i|s_i) \\ &= \sum_i \log \frac{\sum_h e^{\mathbf{f}(s_i,h,t_i)\cdot\Theta}}{\sum_{t,h} e^{\mathbf{f}(s_i,h,t)\cdot\Theta}} \quad (2)\end{aligned}$$

This function is not convex due to the summation over the hidden variable h. To optimize it, we take its partial derivative with respect to θ_j:

$$\begin{aligned}\frac{\partial\mathcal{L}(\Theta)}{\partial\theta_j} &= \sum_i \mathbb{E}_{p_\Theta(h|s_i,t_i)}[f_j(s_i, h, t_i)] \\ &- \sum_i \mathbb{E}_{p_\Theta(t,h|s_i)}[f_j(s_i, h, t)] \quad (3)\end{aligned}$$

which requires computation of expectations terms under two different distributions. Such statistics can be collected efficiently with a forward-backward style algorithm in polynomial time (Okanohara et al., 2006). We will discuss the time complexity for our case in the next section.

Given its partial derivatives in Equation 3, one could optimize the objective function of Equation 2 with stochastic gradient ascent (LeCun et al., 1998) or L-BFGS (Liu and Nocedal, 1989). We choose to use L-BFGS for all our experiments in this paper.

Inference involves computing the most probable template realization t for a given event span:

$$\arg\max_t P_\Theta(t|s) = \arg\max_t \sum_h P_\Theta(t, h|s) \quad (4)$$

where the possible hidden assignments h need to be marginalized out. In this task, a particular realization t already uniquely defines a particular segmentation (mention boundaries) of the event span, thus the h only contributes type information to t. As we will discuss in Section 2.3, only a collection of local features are defined. Thus, a Viterbi-style dynamic programming algorithm is used to efficiently compute the desired solution.

2.1 Possible Segmentations

According to Equation 3, summing over all possible h is required. Since one primary assumption is that we have access to the output of existing mention identification and typing systems, the set of all possible mentions defines a lattice representation containing the set of all possible segmentations that comply with such mention-level information. Assuming there are A possible arguments for the event and K annotated mentions, the complexity of the forward-backward style algorithm is in $O(A^3K^2)$ under the "second-order" setting that we will discuss in Section 2.2. Typically, K is smaller than the number of words in the span, and the factor A^3 can be regarded as a constant. Thus, the algorithm is very efficient.

As we have mentioned earlier, such coarse information, as produced by existing resources, could be highly ambiguous and noisy. Also, the output mentions can highly overlap with each other. For example, the phrase "*North Korea*" as in "*North Korea's military*" can be assigned both type GPE and LOC, while "*North Korea's military*" can be assigned the type ORG. Our model will need to disambiguate the mention boundaries as well as their types.

2.2 The Gap Segments

We believe the gap segments[2] are important to model since they can potentially capture dependencies between two or more adjacent arguments. For example, the word sequence "*may have fired*" clearly indicates an Attacker-Instrument relation between the two mentions "*North Korea's military*" and "*a laser*". Since we are only interested in modeling dependencies between adjacent argument segments, we assign hard labels to each gap segment based on its contextual argument information. Specifically, the label of each gap segment is uniquely determined by its surrounding argument segments with a list representation. For example, in a "first-order" setting, the gap segment that appears between its previous argument segment "ATTACKER" and its next argument segment "INSTRUMENT" is annotated as the list consisting of two elements: [ATTACKER, INSTRUMENT]. To capture longer-range dependencies, in this work we use a "second-order" setting (as shown in Figure 2),

[2]The length of a gap segment is arbitrary (including zero), unlike the seminal semi-Markov CRF model of Sarawagi and Cohen (2004).

which means each gap segment is annotated with a list that consists of its previous two argument segments as well as its subsequent one.

2.3 Features

Feature functions are factorized as products of two indicator functions: one defined on the input sequence (input features) and the other on the output labels (output features). In other words, we could re-write $f_j(s, h, t)$ as $f_k^{in}(s) \times f_l^{out}(h, t)$.

For gap segments, we consider the following input feature templates:

- N-GRAM: Indicator function for n-gram appeared in the segment ($n = 1, 2$)
- ANCHOR: Indicator function for its relative position to the event anchor words (to the left, to the right, overlaps, contains)

and the following output feature templates:

- 1STORDER: Indicator function for the combination of its immediate left argument and its immediate right argument.
- 2NDORDER: Indicator function for the combination of its immediate two left arguments and its immediate right argument.

For argument segments, we also define the same input feature templates as above, with the following additional ones to capture contextual information:

- CWORDS: Indicator function for the previous and next k ($= 1, 2, 3$) words.
- CPOS: Indicator function for the previous and next k ($= 1, 2, 3$) words' POS tags.

and we define the following output feature template:

- ARGTYPE: Indicator function for the combination of the argument and its associated type.

Although the semi-Markov CRF model gives us the flexibility in introducing features that can not be exploited in a standard CRF, such as entity name similarity scores and distance measures, in practice we found the above simple and general features work well. This way, the unnormalized score assigned to each structure is essentially a linear sum of the feature weights, each corresponding to an indicator function.

3 Learning without Annotated Data

The supervised model presented in the previous section requires substantial human efforts to annotate the training instances. Human annotations can be very expensive and sometimes impractical. Even if annotators are available, getting annotators to agree with each other is often a difficult task in itself. Worse still, annotations often can not be reused: experimenting on a different domain or dataset typically require annotating new training instances for that particular domain or dataset.

We investigate inexpensive methods to alleviate this issue in this section. We introduce a novel general learning framework called *structured preference modeling*, which allows arbitrary prior knowledge about structures to be introduced to the learning process in a declarative manner.

3.1 Structured Preference Modeling

Denote by $\mathcal{X}_\Omega$ and $\mathcal{Y}_\Omega$ the entire input and output space, respectively. For a particular input $x \in \mathcal{X}_\Omega$, the set $x \times \mathcal{Y}_\Omega$ gives us all possible structures that contain x. However, structures are not equally good. Some structures are generally regarded as better structures while some are worse.

Let's asume there is a function $\kappa : \{x \times \mathcal{Y}_\Omega \rightarrow [0, 1]\}$ that measures the quality of the structures. This function returns the quality of a certain structure (x, y), where the value 1 indicates a perfect structure, and 0 an impossible structure.

Under such an assumption, it is easy to observe that for a good structure (x, y), we have $p_\Theta(x, y) \times \kappa(x, y) = p_\Theta(x, y)$, while for a bad structure (x, y), we have $p_\Theta(x, y) \times \kappa(x, y) = 0$.

This motivates us to optimize the following objective function:

$$\mathcal{L}_u(\Theta) = \sum_i \log \frac{\sum_y p_\Theta(x_i, y) \times \kappa(x_i, y)}{\sum_y p_\Theta(x_i, y)} \quad (5)$$

Intuitively, optimizing such an objective function is equivalent to pushing the probability mass from bad structures to good structures corresponding to the same input.

When the preference function κ is defined as the indicator function for the correct structure (x_i, y_i), the numerator terms of the above formula are simply of the forms $p_\Theta(x_i, y_i)$, and the model corresponds to the fully supervised CRF model.

The model also contains the latent-variable CRF as a special case. In a latent-variable CRF, we have input-output pairs (x_i, y_i), but the underlying specific structure h that contains both x_i and y_i is hidden. The objective function is:

$$\sum_i \log \frac{\sum_h p_\Theta(x_i, h, y_i)}{\sum_{h,y'} p_\Theta(x_i, h, y')} \quad (6)$$

where $p_\Theta(x_i, h, y_i) = 0$ unless h contains (x_i, y_i). We define the following two functions:

$$q_\Theta(x_i, h) = \sum_{y'} p_\Theta(x_i, h, y') \quad (7)$$

$$\kappa(x_i, h) = \begin{cases} 1 & h \text{ contains } (x_i, y_i) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Note that this definition of κ models instance-specific preferences since it relies on y_i, which can be thought of as certain external prior knowledge related to x_i. It is easy to verify that $p_\Theta(x_i, h, y_i) = q_\Theta(x_i, h) \times \kappa(x_i, h)$, with q_Θ remains a distribution. Thus, we could re-write the objective function as:

$$\sum_{i=1} \log \frac{\sum_h q_\Theta(x_i, h) \times \kappa(x_i, h)}{\sum_h q_\Theta(x_i, h)} \quad (9)$$

This shows that the latent-variable CRF is a special case of our objective function, with the above-defined κ function. Thus, this new objective function of Equation 5 is a generalization of both the supervised CRF and the latent-variable CRF.

The preference function κ serves as a source from which certain prior knowledge about the structure can be injected into our model in a principled way. Note that the function is defined at the complete structure level. This allows us to incorporate both local and arbitrary global structured information into the preference function.

Under the log-linear parameterization, we have:

$$\mathcal{L}'(\Theta) = \sum_i \log \frac{\sum_y e^{\mathbf{f}(x_i, y)\cdot\Theta} \times \kappa(x_i, y)}{\sum_y e^{\mathbf{f}(x_i, y)\cdot\Theta}} \quad (10)$$

This is again a non-convex optimization problem in general, and to solve it we take its partial derivative with respect to θ_k:

$$\frac{\partial \mathcal{L}'(\Theta)}{\partial \theta_k} = \sum_i \mathbb{E}_{p_\Theta(y|x_i;\kappa)}[f_k(x_i, y)] - \sum_i \mathbb{E}_{p_\Theta(y|x_i)}[f_k(x_i, y)] \quad (11)$$

$$p_\Theta(y|x_i;\kappa) \propto e^{\mathbf{f}(x_i, y)\cdot\Theta} \times \kappa(x_i, y)$$

$$p_\Theta(y|x_i) \propto e^{\mathbf{f}(x_i, y)\cdot\Theta}$$

3.2 Approximate Learning

Computation of the denominator terms of Equation 10 (and the second term of Equation 11) can be done efficiently and exactly with dynamic programming. Our main concern is the computation of its numerator terms (and the first term of Equation 11).

The preference function κ is defined at the complete structure level. Unless the function is defined in specific forms that allow tractable dynamic programming (in the supervised case, which gives a unique term, or in the hidden variable case, which can define a packed representations of derivations), the efficient dynamic programming algorithm used by CRF is no longer generally applicable for arbitrary κ. In general, we resort to approximations.

In this work, we exploit a specific form of the preference function κ. We assume that there exists a projection from another decomposable function to κ. Specifically, we assume a collection of auxiliary functions, each of the form $\kappa_p : (x, y) \rightarrow R$, that scores a property p of the complete structure (x, y). Each such function measures certain aspect of the quality of the structure. These functions assign positive scores to good structural properties and negative scores to bad ones. We then define $\kappa(x, y) = 1$ for all structures that appear at the top-n positions as ranked by $\sum_p \kappa_p(x, y)$ for all possible y's, and $\kappa(x, y) = 0$ otherwise. We show some actual κ_p functions used for a particular event in Section 5.

At each iteration of the training process, to generate such a n-best list, we first use our model to produce top $n \times b$ candidate outputs as scored by the current model parameters, and extract the top n outputs as scored by $\sum_p \kappa_p(x, y)$. In practice we set $n = 10$ and $b = 1000$.

3.3 Event Extraction

Now we can obtain the objective function for our event extraction task. We replace x by s and y by (h, t) in Equation 10. This gives us the following function:

$$\mathcal{L}_u(\Theta) = \sum_i \log \frac{\sum_{t,h} e^{\mathbf{f}(s_i, h, t)\cdot\Theta} \times \kappa(s_i, h, t)}{\sum_{t,h} e^{\mathbf{f}(s_i, h, t)\cdot\Theta}} \quad (12)$$

The partial derivatives are as follows:

$$\frac{\partial \mathcal{L}_u(\Theta)}{\partial \theta_k} = \sum_i \mathbb{E}_{p_\Theta(t,h|s_i;\kappa)}[f_k(s_i, h, t)] - \sum_i \mathbb{E}_{p_\Theta(t,h|s_i)}[f_k(s_i, h, t)] \quad (13)$$

$$p_\Theta(t, h|s_i;\kappa) \propto e^{\mathbf{f}(s_i, h, t)\cdot\Theta} \times \kappa(s_i, h, t)$$

$$p_\Theta(t, h|s_i) \propto e^{\mathbf{f}(s_i, h, t)\cdot\Theta}$$

Recall that s is an event span, t is a specfic realization of the event template, and h is the hidden mention information for the event span.

4 Discussion: Preferences v.s. Constraints

Note that the objective function in Equation 5, if written in the additive form, leads to a cost function reminiscent of the one used in constraint-driven learning algorithm (CoDL) (Chang et al., 2007) (and similarly, posterior regularization (Ganchev et al., 2010), which we will discuss later at Section 6). Specifically, in CoDL, the following cost function is involved in its EM-like inference procedure:

$$\arg\max_{y} \Theta \cdot \mathbf{f}(x, y) - \rho \sum_{c} d(y, \mathcal{Y}_c) \qquad (14)$$

where $\mathcal{Y}_c$ defines the set of y's that all satisfy a certain constraint c, and d defines a distance function from y to that set. The parameter ρ controls the degree of the penalty when constraints are violated.

There are some important distinctions between structured preference modeling (PM) and CoDL. CoDL primarily concerns *constraints*, which penalizes bad structures without explicitly rewarding good ones. On the other hand, PM concerns *preferences*, which can explicitly reward good structures.

Constraints are typically useful when one works on structured prediction problems for data with certain (often rigid) regularities, such as citations, advertisements, or POS tagging for complete sentences. In such tasks, desired structures typically present certain canonical forms. This allows declarative constraints to be specified as either local structure prototypes (e.g., in citation extraction, the word *pp*. always corresponds to the PAGES field, while *proceedings* is always associated with BOOKTITLE or JOURNAL), or as certain global regulations about complete structures (e.g., at least one word should be tagged as verb when performing a sentence-level POS tagging).

Unfortunately, imposing such (hard or soft) constraints for certain tasks such as ours, where the data tends to be of arbitrary forms without many rigid regularities, can be difficult and often inappropriate. For example, there is no guarantee that a certain argument will always be present in the event span, nor should a particular mention, if appeared, always be selected and assigned to a specific argument. For example, in the example event span given in Section 1, both "*March*" and "*Tuesday*" are valid candidate mentions for the TIME-WITHIN argument given their annotated type TME. One important clue is that *March* appears after the word *in* and is located nearer to other mentions that can be potentially useful arguments. However, encoding such information as a general constraint can be inappropriate, as potentially better structures can be found if one considers other alternatives. On the other hand, if we believe the structural pattern "*at* TARGET *in* TIME-WITHIN" is in general considered a better sub-structure than "*said* TIME-WITHIN" for the "Attack" event, we may want to assign structured preference to a complete structure that contains the former, unless there exist other structured evidence showing the latter turns out to be better.

In this work, our preference function is related to another function that can be decomposed into a collection of property functions κ_p. Each of them scores a certain aspect of the complete structure. This formulation gives us a complete flexibility to assign arbitrary structured preferences, where positive scores can be assigned to good properties, and negative scores to bad ones. Thus, in this way, the quality of a complete structure is jointly measured with multiple different property functions.

To summarize, preferences are an effective way to "define" the event structure to the learner, which is essential in an unsupervised setting, which may not be easy to do with other forms of constraints. Preferences are naturally decomposable, which allows us to extend their impact without significantly effecting the complexity of inference.

5 Experiments

In this section, we present our experimental results on the standard ACE05[3] dataset (newswire portion). We choose to perform our evaluations on 4 events (namely, "Attack", "Meet", "Die" and "Transport"), which are the only events in this dataset that have more than 50 instances. For each event, we randomly split the instances into two portions, where 70% are used for learning, and the remaining 30% for evaluation. We list the corpus statistics in Table 2.

To present general results while making minimal assumptions, our primary event extraction results

[3] http://www.itl.nist.gov/iad/mig/tests/ace/2005/doc/

Event	Without Annotated Training Data				With Annotated Training Data			
	Random	Unsup	Rule	PM	MaxEnt-b	MaxEnt-t	MaxEnt-p	semi-CRF
Attack	20.47	30.12	39.25	42.02	54.03	58.82	**65.18**	63.11
Meet	35.48	26.09	44.07	63.55	65.42	70.48	75.47	**76.64**
Die	30.03	13.04	40.58	55.38	51.61	59.65	63.18	**67.65**
Transport	20.40	6.11	44.34	57.29	53.76	57.63	61.02	**64.19**

Table 1: Performance for different events under different experimental settings, with gold mention boundaries and types. We report F1-measure percentages.

Event	#A	Learning Set		Evaluation Set		**#P**
		#I	#M	#I	#M	
Attack	8	188	300/509	78	121/228	**7**
Meet	7	57	134/244	24	52/98	**7**
Die	9	41	89/174	19	33/61	**6**
Transport	13	85	243/426	38	104/159	**6**

Table 2: Corpus statistics (#A: number of possible arguments for the event; #I: number of instances; #M: number of active/total mentions; **#P: number of preference patterns** used for performing our structured preference modeling.)

are independent of mention identification and typing modules, which are based on the gold mention information as given by the dataset. Additionally, we present results obtained by exploiting our in-house automatic mention identification and typing module, which is a hybrid system that combines statistical and rule-based approaches. The module's statistical component is trained on the ACE04 dataset (newswire portion) and overall it achieves a micro-averaged F1-measure of 71.25% at our dataset.

5.1 With Annotated Training Data

With hand-annotated training data, we are able to train our model in a fully supervised manner. The right part of Table 1 shows the performance for the fully supervised models. For comparison, we present results from several alternative approaches based a collection of locally trained maximum entropy (MaxEnt) classifiers. In these approaches, we treat each argument of the template as one possible output class, plus a special "NONE" class for not selecting it as an argument. We train and apply the classifiers on argument segments (i.e., mentions) only. All the models are trained with the same feature set used in the semi-CRF model.

In the simplest baseline approach MaxEnt-b, type information for each mention is simply treated as one special feature. In the approach MaxEnt-t, we instead use the type information to constrain the classifier's predictions based on the acceptable types associated with each argument. This approach gives better performance than that of MaxEnt-b. This indicates that such locally trained classifiers are not robust enough to disambiguate arguments that take different types. As such, type information serving as additional constraints at the end does help.

To assess the importance of structured preference, we also perform experiments where structured preference information is incorporated at the inference time of the MaxEnt classifiers. Specifically, for each event, we first generate n-best lists for output structures. Next, we re-rank this list based on scores from our structured preference functions (we used the same preferences as to be discussed in the next section). The results for these approaches are given in the column of MaxEnt-p of Table 1. This simple approach gives us significant improvements, closing the gap between locally trained classifiers and the joint model (in one case the former even outperforms the latter). Note that no structured preference information is used when training and evaluating our semi-CRF model. This set of results is not surprising. In fact, similar observations are also reported in previous works when comparing joint model against local models with constraints incorporated (Roth and Yih, 2005). This clearly indicates that structured preference information is crucial to model.

5.2 Without Annotated Training Data

Now we turn to experiments for the more realistic scenario where human annotations are not available.

We first build our simplest baseline by randomly assigning arguments to each mention with mention type information serving as constraints. Averaged results over 1000 runs are reported in the first column of Table 1.

Since our model formulation leaves us with complete freedom in designing the preference function,

Type	Preference pattern (p)
General	{*at*|*in*|*on*} followed by PLACE
	{*during*|*at*|*in*|*on*} followed by TIME-WITHIN
Die	AGENT (immediately) followed by {*killed*}
	{*killed*} (immediately) followed by VICTIM
	VICTIM (immediately) followed by {*be killed*}
	AGENT followed by {*killed*} (immediately) followed by VICTIM
Transport	X immediately followed by {,|*and*} immediately followed by X, where X $\in$ {ORIGIN|DESTINATION}
	{*from*|*leave*} (immediately) followed by ORIGIN
	{*at*|*in*|*to*|*into*} immediately followed by DESTINATION
	PERSON followed by {*to*|*visit*|*arrived*}

Figure 3: The **complete list** of preference patterns used for the "Die" and "Transport" event. We simply set $\kappa_p = 1.0$ for all p's. In other words, when a structure contains a pattern, its score is incremented by 1.0. We use {} to refer to a set of possible words or arguments. For example, {*from*|*leave*} means a word which is either *from* or *leave*. The symbol () denotes optional. For example, "{*killed*} (immediately) followed by VICTIM" is equivalent to the following two preferences: "{*killed*} immediately followed by VICTIM", and "{*killed*} followed by VICTIM".

one could design arbitrarily good, domain-specific or even instance-specific preferences. However, to demonstrate its general effectiveness, in this work we only choose a minimal amount of general preference patterns for evaluations.

We make our preference patterns as general as possible. As shown in the last column (#P) of Table 2, we use only 7 preference patterns each for the "Attack" and "Meet" events, and 6 patterns each for the other two events. In Figure 3, we show the complete list of the 6 preference patterns for the "Die" and "Transport" event used for our experiments. Out of those 6 patterns, 2 are more general patterns shared across different events, and 4 are event-specific. In contrast, for example, for the "Die" event, the supervised approach requires human to select from 174 candidate mentions and annotate 89 of them.

Despite its simplicity, it works very well in practice. Results are given in the column of "PM" of Table 1. It generally gives competitive performance as compared to the supervised MaxEnt baselines.

On the other hand, a completely unsupervised approach where structured preferences are not specified, performs substantially worse. To run such completely unsupervised models, we essentially follow the same training procedure as that of the preference modeling, except that structured preference information is not in place when generating the n-best list. In the absence of proper guidances, such a procedure can easily converge to bad local minima. The results are reported in the "Unsup" column of Table 1. In practice, we found that very often, such a model would prefer short structures where many mentions are not selected as desired. As a result, the unsupervised model without preference information can even perform worse than the random baseline [4].

Finally, we also compare against an approach that regards the preferences as rules. All such rules are associated with a same weight and are used to jointly score each structure. We then output the structure that is assigned the highest total weight. Such an approach performs worse than our approach with preference modeling. The results are presented in the column of "Rule" of Table 1. This indicates that our model is able to learn to generalize with features through the guidance of our informative preferences. However, we also note that the performance of preference modeling depends on the actual quality and amount of preferences used for learning. In the extreme case, where only few preferences are used, the performance of preference modeling will be close to that of the unsupervised approach, while the rule-based approach will yield performance close to that of the random baseline.

The results with automatically predicted mention boundaries and types are given in Table 3. Similar observations can be made when comparing the performance of preference modeling with other approaches. This set of results further confirms the effectiveness of our approach using preference modeling for the event extraction task.

6 Related Work

Structured prediction with limited supervision is a popular topic in natural language processing.

[4] For each event, we only performed 1 run with all the initial feature weights set to zeros.

Event	Random	Unsup	PM	semi-CRF
Attack	14.26	26.19	32.89	**46.92**
Meet	26.65	14.08	45.28	**58.18**
Die	19.17	9.09	44.44	**48.57**
Transport	15.78	10.14	49.73	**52.34**

Table 3: Event extraction performance with automatic mention identifier and typer. We report F1 percentage scores for preference modeling (PM) as well as two baseline approaches. We also report performance of the supervised approach trained with the semi-CRF model for comparison.

Prototype driven learning (Haghighi and Klein, 2006) tackled the sequence labeling problem in a primarily unsupervised setting. In their work, a Markov random fields model was used, where some local constraints are specified via their *prototype list*.

Constraint-driven learning (CoDL) (Chang et al., 2007) and posterior regularization (PR) (Ganchev et al., 2010) are both primarily semi-supervised models. They define a constrained EM framework that regularizes posterior distribution at the E-step of each EM iteration, by pushing posterior distributions towards a constrained posterior set. We have already discussed CoDL in Section 4 and gave a comparison to our model. Unlike CoDL, in the PR framework constraints are relaxed to *expectation constraints*, in order to allow tractable dynamic programming. See also Samdani et al. (2012) for more discussions.

Contrastive estimation (CE) (Smith and Eisner, 2005a) is another log-linear framework for primarily unsupervised structured prediction. Their objective function is related to the pseudolikelihood estimator proposed by Besag (1975). One challenge is that it requires one to design a priori an effective neighborhood (which also needs to be designed in certain forms to allow efficient computation of the normalization terms) in order to obtain optimal performance. The model has been shown to work in unsupervised tasks such as POS induction (Smith and Eisner, 2005a), grammar induction (Smith and Eisner, 2005b), and morphological segmentation (Poon et al., 2009), where good neighborhoods can be identified. However, it is less intuitive what constitutes a good neighborhood in this task.

The neighborhood assumption of CE is relaxed in another latent structure approach (Chang et al., 2010a; Chang et al., 2010b) that focuses on semi-supervised learning with indirect supervisions, inspired by the CoDL model described above.

The locally normalized logistic regression (Berg-Kirkpatrick et al., 2010) is another recently proposed framework for unsupervised structured prediction. Their model can be regarded as a generative model whose component multinomial is replaced with a miniature logistic regression where a rich set of local features can be incorporated. Empirically the model is effective in various unsupervised structured prediction tasks, and outperforms the globally normalized model. Although modeling the semi-Markov properties of our segments (especially the gap segments) in our task is potentially challenging, we plan to investigate in the future the feasibility for our task with such a framework.

7 Conclusions

In this paper, we present a novel model based on the semi-Markov conditional random fields for the challenging event extraction task. The model takes in coarse mention boundary and type information and predicts complete structures indicating the corresponding argument role for each mention.

To learn the model in an unsupervised manner, we further develop a novel learning approach called *structured preference modeling* that allows structured knowledge to be incorporated effectively in a declarative manner.

Empirically, we show that knowledge about structured preference is crucial to model and the preference modeling is an effective way to guide learning in this setting. Trained in a primarily unsupervised manner, our model incorporating structured preference information exhibits performance that is competitive to that of some supervised baseline approaches. Our event extraction system and code will be available for download from our group web page.

Acknowledgments

We would like to thank Yee Seng Chan, Mark Sammons, and Quang Xuan Do for their help with the mention identification and typing system used in this paper. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, AFRL, or the US government.

References

T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *Proc. of HLT-NAACL'10*, pages 582–590.

J. Besag. 1975. Statistical analysis of non-lattice data. *The Statistician*, pages 179–195.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of ACL'07*, pages 280–287.

M. Chang, D. Goldwasser, D. Roth, and V. Srikumar. 2010a. Discriminative learning over constrained latent representations. In *Proc. of NAACL'10*, 6.

M. Chang, V. Srikumar, D. Goldwasser, and D. Roth. 2010b. Structured output learning with indirect supervision. In *Proc. ICML'10*.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research (JMLR)*, 11:2001–2049.

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *Proc. of HLT-NAACL'06*, pages 320–327.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML'01*, pages 282–289.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, pages 2278–2324.

D.C. Liu and J. Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.

D. Okanohara, Y. Miyao, Y. Tsuruoka, and J. Tsujii. 2006. Improving the scalability of semi-markov conditional random fields for named entity recognition. In *Proc. of ACL'06*, pages 465–472.

H. Poon, C. Cherry, and K. Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proc. of HLT-NAACL'09*, pages 209–217.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of CoNLL'09*, pages 147–155.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proc. of ACL-HLT'11*, pages 1375–1384.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *Proc. of ICML'05*, pages 736–743.

R. Samdani, M. Chang, and D. Roth. 2012. Unified expectation maximization. In *Proc. NAACL'12*.

S. Sarawagi and W.W. Cohen. 2004. Semi-markov conditional random fields for information extraction. *NIPS'04*, pages 1185–1192.

N.A. Smith and J. Eisner. 2005a. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL'05*, pages 354–362.

N.A. Smith and J. Eisner. 2005b. Guiding unsupervised grammar induction using contrastive estimation. In *Proc. of IJCAI Workshop on Grammatical Inference Applications*, pages 73–82.

J. Strötgen and M. Gertz. 2010. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proc. of SemEval'10*, pages 321–324.