Constrained Multi-Task Learning for Event Coreference Resolution

Jing Lu and Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{ljwinnie,vince}@hlt.utdallas.edu

Abstract

We propose a neural event coreference model in which event coreference is jointly trained with five tasks: trigger detection, entity coreference, anaphoricity determination, realis detection, and argument extraction. To guide the learning of this complex model, we incorporate cross-task consistency constraints into the learning process as soft constraints via designing penalty functions. In addition, we propose the novel idea of viewing entity coreference and event coreference as a single coreference task, which we believe is a step towards a unified model of coreference resolution. The resulting model achieves state-of-the-art results on the KBP 2017 event coreference dataset.

1 Introduction

Event coreference resolution is the task of determining whether two event mentions in a document refer to the same real-world event. For two event mentions to be coreferent, their *triggers* (i.e., the words realizing the occurrence of the events) should have the same *subtype* and their corresponding *arguments* (e.g., the times, places, and people involved) have to be entity-coreferent. However, identifying potential arguments (which is performed by an entity extraction system), linking arguments to their event mentions (which is also performed by an event extraction system), and determining whether two event arguments are coreferent (which is the job of an entity coreference resolver), are all nontrivial tasks. Hence, a key challenge in designing an event coreference resolver involves determining how to integrate these noisy components.

One of the most common approaches to event coreference resolution is *pipelined* approaches, where a *trigger detection* component, which identifies triggers and assigns event subtypes to them, is followed by an *event coreference* component, which clusters coreferent event mentions. It should therefore not be surprising that errors propagate from the trigger detection component to the event coreference component. To avoid aggravating this error propagation problem, knowledge provided by other information extraction (IE) components (e.g., entity coreference, event arguments) is typically employed as features for training event coreference models (Chen et al., 2009; McConky et al., 2012; Cybulska and Vossen, 2013; Araki et al., 2014; Liu et al., 2014; Peng et al., 2016; Krause et al., 2016; Choubey and Huang, 2017). Oftentimes, these features provide limited improvements to event coreference models as they are too noisy to be useful.

Though less popular than pipelined approaches, *bootstrapping* approaches have been used for event coreference resolution, where an event coreference model is bootstrapped with models trained for one or more related IE tasks. For instance, Lee et al. (2012) incrementally build clusters of coreferent event and entity mentions by iteratively bootstrapping event coreference output using entity coreference output and vice versa. While in pipelined approaches only upstream tasks can influence downstream tasks, in bootstrapping approaches different tasks can influence each other. Nevertheless, errors made in earlier iterations of the bootstrapping process cannot be undone in later iterations.

Joint learning approaches have recently emerged as promising approaches to event coreference owing to their ability to address error propagation. In these approaches, two or more tasks are *jointly* trained. For instance, Araki and Mitamura (2015) learn a joint model for trigger detection and event coreference using a structured perceptron, and Lu and Ng (2017) learn a joint model for trigger detection, event coreference, and anaphoricity determination using a structured conditional random field. The key advantage of these models is that the tasks involved can benefit from each other during training. However, since a jointly learned model involves multiple tasks, it is typically complex. In fact, it is by no means easy to scale such a model

to a large number of tasks because of the high computational complexity involved in learning.

Joint inference approaches have also been applied to event coreference resolution. For instance, Chen and Ng (2016) and Lu et al. (2016) first train separate models for entity coreference, trigger detection, argument extraction, and event coreference, then use Integer Linear Programming or Markov Logic Networks to jointly infer the outputs of these tasks subject to (mostly) hard cross-task consistency constraints. For instance, one such hard constraint says that two coreferent event mentions should have the same event subtype. Since the models are trained independently, they cannot benefit from each other and could be noisy. Worse still, performing joint inference using hard constraints over (very) noisy outputs could do more harm than good. For instance, if two event mentions are correctly classified as coreferent but one of their subtypes is misclassified, then enforcing the aforementioned constraint might cause the joint inference procedure to incorrectly infer that the two are not coreferent. This explains why joint inference approaches have become less popular than joint learning approaches in recent years.

In light of the above discussion, we seek to advance the state of the art in event coreference resolution by proposing a model that jointly learns *six* tasks: trigger detection, event coreference, entity coreference, anaphoricity determination, argument extraction, and realis detection. As noted above, joint learning typically presents a serious computational challenge, and training a complex joint model involving six tasks would not have been possible without the advent of the neural NLP era.

While multi-task learning in a neural network typically allows the different tasks involved to benefit from each other via learning shared representations, we hypothesize that the model would benefit additional guidance given that the learning task, which involves six tasks, is so complex. Consequently, we propose to guide the learning process by exploiting cross-task consistency constraints. As mentioned above, such consistency constraints are typically employed in joint inference and rarely in joint learning. Moreover, unlike in joint inference where such constraints are typically implemented as hard constraints, we provide flexibility by implementing them as soft constraints. Specifically, we design penalty functions for penalizing outputs that violate a constraint, where the degree of penalty depends on the extent of the violation.

Another contribution of our work involves proposing the idea of a *unified* coreference model. So far, entity and event coreference have always been viewed as two separate tasks, where links between entity mentions are distinguished from links between event mentions. However, their similarity has led us to hypothesize that they could be viewed as a single task, where coreference links are established between a set of mentions without distinguishing between entity and event mentions.

2 Related Work

Traditional resolvers. Many existing event coreference resolvers, including those that employ the four approaches described in the introduction, are developed in the pre-neural era. resolvers. For a detailed overview of these non-neural resolvers and the wide variety of hand-engineered features they employ, we refer the reader to Lu and Ng (2018).

Neural resolvers. Of particular relevance to our work are neural event coreference models (e.g., Nguyen et al. (2016), Choubey and Huang (2017, 2018), Huang et al. (2019)). Unlike their traditional counterparts, neural coreference models can leverage the knowledge learned from large unlabeled corpora through pretrained word embeddings or transfer learning. Existing neural event coreference models are pipeline-based and seek to learn *word* representations so that coreferent event mentions have similar word embeddings, effectively making the rather unrealistic assumption that an event trigger is composed of a single token (Nguyen et al., 2016). In contrast, our neural resolver is a joint model and seeks to learn the representations of text spans, each of which corresponds to a candidate event trigger and may be composed of more than one token, so that coreferent event mentions have similar span representations.

Constrained learning in neural models. Another line of related work concerns the use of constraints in neural models (Li et al., 2019; Wang et al., 2020), where constraints are represented as first order logic formulas and compiled into the loss functions. These models are typically trained to minimize the weighted sum of task losses and constraint losses. Rather than introduce additional terms in the loss function, we employ constraints as penalty terms when learning to score how likely two event mentions are coreferent, effectively making the two mentions less likely to be coreferent if

{Two men}$_{en1}$ accused of {hacking}$_{ev1}$ {a British soldier}$_{en2}$ to {death}$_{ev2}$ last month appeared in {separate courts}$_{en3}$ for hearings. {The men}$_{en4}$, {Michael Adebolajo}$_{en5}$, 28, and {Michael Adebowale}$_{en6}$, 22, face {murder}$_{ev3}$ charges. {Adebolajo}$_{en7}$ was also charged with other offenses, including the attempted {murder}$_{ev4}$ of {two police officers}$_{en8}$.

Table 1: Event coreference example.

a constraint is violated.

3 Definitions

Before formally defining the six tasks in the next section, we introduce several related definitions.

- An **event mention** is an explicit occurrence of an event consisting of a textual trigger, arguments or participants (if any), and the event *subtype*, and can optionally be characterized by a set of *attributes* and their values.
- An **entity mention** is an explicit mention of an entity in a text that has an entity type.
- An **event trigger** is a string of text that most clearly expresses the occurrence of an event, usually a word or a multi-word phrase.
- An **event argument** is an argument filler that plays a certain *role* in an event.
- **Realis** denotes whether an event actually happened or will happen in the future, or whether it is a generic event. Its value can be ACTUAL, GENERIC or OTHER.
- An **event/entity coreference chain** is a group of event/entity mentions that refer to the same real-world event/entity.

To better understand these definitions, consider the example in Table 1. The four event mentions ($ev1$, $ev2$, $ev3$, $ev4$) are triggered by "hacking", "death", "murder", and "murder" respectively. The first three have ACTUAL as their realis and the last one belongs to OTHER. While $ev2$ has LIFE_DIE as its subtype, the remaining ones all have subtype CONFLICT_ATTACK. Among the eight entity mentions ($en1, \ldots, en8$), $en3$ has FACILITY as its type and the remaining ones are all PERSONs. $en1$ and $en2$ are the arguments of $ev1$ filling the roles of ATTACKER and TARGET respectively, whereas $en4$ is the argument of $ev3$ having the role ATTACKER. There are two entity coreference chains (one composed of $en1$ and $en4$ and the other $en5$ and $en7$) and one event coreference chain ($ev1$ and $ev3$).

4 Model

We design a *span-based* neural model for event coreference resolution owing to its ability to effectively learn representations of text *spans*. While span-based models have been successfully applied to a variety of entity-based IE tasks such as entity coreference (Lee et al., 2017; Joshi et al., 2020) and relation extraction (Luan et al., 2019), they have not been applied to event coreference.

More formally, our model takes as input a document D represented as a sequence of word tokens, from which we extract all possible intra-sentence spans of up to length L. It simultaneously learns six tasks, which we define below.

The *trigger detection* task aims to assign each span i a subtype label y_i. Each y_i takes a value in a subtype inventory or NONE, which indicates that i is not a trigger. The model predicts i's subtype to be $y_i^* = \arg\max_{y_t} s_t(i, y_t)$, where s_t is a scoring function suggesting i's likelihood of having y_i as its subtype.

The *event coreference resolution* task aims to assign span i an antecedent y_c, where $y_c \in \{1, \ldots, i-1, \epsilon\}$. In other words, the value of y_c is the id of i's antecedent, which can be one of the preceding spans or a dummy antecedent ϵ (if the event mention underlying i starts a new cluster). We define the following scoring function:

$$s_c(i,j) = \begin{cases} 0 & j = \epsilon \\ s_m(i) + s_m(j) + s_p(i,j) & j \neq \epsilon \end{cases} \quad (1)$$

where $s_m(i)$ is the score suggesting span i's likelihood of being a trigger and $s_p(i,j)$ is a pairwise coreference score computed over span i and a preceding span j. The model predicts the antecedent of i to be $y_c^* = \arg\max_{j \in \mathcal{Y}(i)} s_c(i,j)$, where $\mathcal{Y}(i)$ is the set of i's candidate antecedents.

The *entity coreference resolution* task involves identifying entity mentions that refer to the same real-world entity. Intuitively, entity coreference is useful for event coreference: two event mentions are not likely to be coreferent if there exists an argument role (e.g., ATTACKER) for which the corresponding arguments in the two event mentions are not entity-coreferent. In our model, it is defined in the same way as the event coreference resolution task except that it operates on the spans identified by the *entity mention detection component* rather than the trigger detection component. The entity mention detection task is defined in the same way

as the trigger detection task except that it aims to assign each span i an entity type label.

The *anaphoricity determination* task aims to assign each span i an anaphoricity label y_a, where y_a can be ANAPHORIC, which indicates that the mention having span i is coreferent with a preceding mention, or NON-ANAPHORIC. The model $s_a(i)$ predicts the mention having span i as anaphoric if and only if $s_a(i) \geq 0$. To train this model, we set the target value to 1 for anaphoric mentions and -1 for non-anaphoric mentions. Anaphoricity is useful for coreference: it prevents non-anaphoric mentions from being resolved.

The *realis detection* task aims to assign each span i a realis label y_r, where $y_r \in \{$ACTUAL, GENERIC, OTHER, ENTITY, and NONE$\}$. As mentioned in Section 3, ACTUAL, GENERIC, and OTHER are labels used for event mention spans. To enable every span i to be assigned a realis label, we augment the realis label set to include ENTITY and NONE. Specifically, ENTITY is a label that is exclusively reserved for spans that correspond to entity mentions, and NONE indicates that i does not correspond to a mention. The model predicts the realis type of i to be $y_r^* = \arg\max_{y_r} s_r(i, y_r)$, where s_r is a scoring function suggesting i's likelihood of having realis type y_r. Realis detection is useful for event coreference: two event mentions cannot be coreferent if their realis labels are different.

The *argument extraction* task aims to assign an argument role label y_o to a candidate argument k of a candidate event mention span i, where (1) k is a candidate entity mention span, and (2) y_o is a role taken from an argument role inventory or NONE, which indicates that the token is not an argument of i. We consider (1) k to be a candidate argument of i if and only if it appears within the same sentence as i; and (2) a span to be a candidate event/entity mention span if it is assigned a non-NONE event/entity type by the Mention Prediction Layer, which we will describe shortly. For each candidate argument k of i, the model predicts its role in i to be $y_o^* = \arg\max_{y_o} s_o(i, k, y_o)$, where s_o is a scoring function suggesting token k's likelihood of being an argument of i having role y_o. Arguments, when combined with entity coreference chains, would be useful for event coreference.

4.1 Model Structure

The model structure, which is shown in Figure 1, is described in detail below.

Span Representation Layer We adapt the independent version of Joshi et al.'s (2019) state-of-the-art entity coreference resolver to event coreference resolution. Specifically, we divide an input document into non-overlapping regions, each of which has size L_d. The word sequence in each region serves as an input training sequence. We then pass the sequence into a pretrained transformer encoder used in SpanBERT-large (Joshi et al., 2020) to encode tokens and their contexts. Finally, we set $\mathbf{g}_i$, the representation of span i, to $[\mathbf{h}_{start(i)}; \mathbf{h}_{end(i)}; \mathbf{h}_{head(i)}; \mathbf{f}_i]$, where $\mathbf{h}_{start(i)}$ and $\mathbf{h}_{end(i)}$ are the hidden vectors of the start and end tokens of the span, $\mathbf{h}_{head(i)}$ is an attention-based head vector and $\mathbf{f}_i$ is a span width feature embedding. To maintain computational tractability, we first compute a score s_m for each span i:

$$s_m(i) = \text{FFNN}_m(\mathbf{g}_i) \tag{2}$$

where FFNN is a standard feedforward neural network. Then we retain only the top $N\%$ of the spans for further processing.

Trigger Prediction Layer For each span i that survives the filtering, we pass its representation $\mathbf{g}_i$ to a FFNN, which outputs a vector $\mathbf{ot}_i$ of dimension T, where T is the number of possible event subtypes (including NONE). $\mathbf{ot}_i(y)$, the yth element of $\mathbf{ot}_i$, is a score indicating i's likelihood of belonging to event subtype y. Specifically:

$$\mathbf{ot}_i = \text{FFNN}_t(\mathbf{g}_i) \tag{3}$$

$$s_t(i, y) = \mathbf{ot}_i(y) \tag{4}$$

Anaphoricity Prediction Layer We predict the anaphoricity value of each top span i as follows. Since the anaphoricity of a mention is dependent on its preceding context, we first concatenate the average of the representations of the 25 tokens immediately preceding i (to approximate i's preceding context) with the span representation $\mathbf{g}_i$. We then pass the resulting vector, $\mathbf{cx}_i$, to a FFNN, which outputs an anaphoricity value. Specifically:

$$s_a(i) = \text{FFNN}_a(\mathbf{cx}_i) \tag{5}$$

Realis Prediction Layer To predict the realis value of each top span i, we pass its representation $\mathbf{g}_i$ to a FFNN, which outputs a vector $\mathbf{or}_i$ of length 5. $\mathbf{or}_i(y)$, the yth element of $\mathbf{or}_i$, is a score indicating i's likelihood of having realis type y:

$$\mathbf{or}_i = \text{FFNN}_r(\mathbf{g}_i) \tag{6}$$

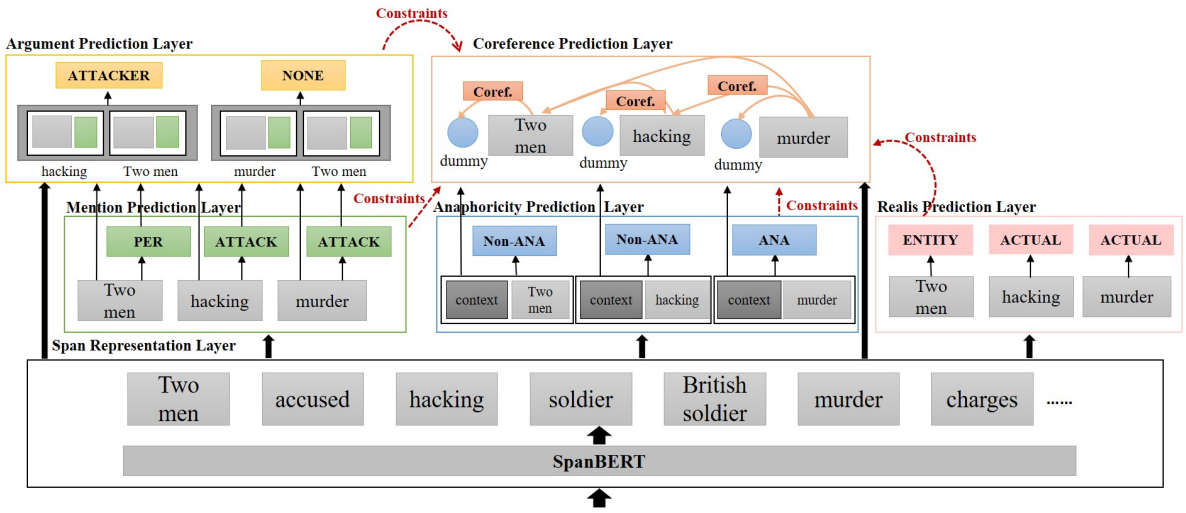

Figure 1: Model structure.

$$s_r(i, y) = \mathbf{or}_i(y) \quad (7)$$

Coreference Prediction Layer To predict event coreference links, we define the pairwise score between span i and span j as follows:

$$s_p(i, j) = \text{FFNN}_c([\mathbf{g}_i; \mathbf{g}_j; \mathbf{g}_i \circ \mathbf{g}_j, \mathbf{u}_{ij}]) \quad (8)$$

where $\circ$ denotes element-wise multiplication, $\mathbf{g}_i \circ \mathbf{g}_j$ encodes the similarity between i and j, and $\mathbf{u}_{ij}$ is a feature embedding encoding the distance between them. We can then compute the full coreference score defined in Equation 1 using Equations 2 and 8. To improve running time, we follow Lee et al. (2018) and use their antecedent pruning method, coarse-to-fine pruning, to reduce the number of candidate antecedents for each anaphor.

Incorporating Entity Coreference The most straightforward way to incorporate entity coreference information into our model would be to have (1) an entity mention detection model that is architecturally identical to the trigger detection model except that it assigns entity type (rather than event subtype) labels to each span, and (2) an entity coreference model that is architecturally identical to the event coreference model described above except that it identifies antecedents for spans provided by the entity mention detection (rather than trigger detection) component. While this would allow entity coreference to interact with event coreference and other tasks via the shared Span Representation Layer, the two coreference tasks would otherwise be learned independently of each other.

Towards the goal of building a *unified* model of coreference, we propose a novel idea: we seek to learn entity and event coreference *simultaneously* by viewing them as a *single* coreference task. From a learning perspective, there is only one task to be learned, which is coreference resolution over a set of mentions. To do so, we extend the Span Representation Layer, the Trigger Prediction Layer, and the Coreference Prediction Layer as follows. First, the Span Representation Layer will identify spans corresponding to *mentions* that are composed of both entity mentions and event mentions even though the model doesn't know (and doesn't need to know) which ones are entity mentions and which ones are event mentions. Second, the Trigger Prediction Layer will assign each mention span a semantic type, which is taken from a type inventory consisting of both entity types and event subtypes (and NONE, if the span is not a mention). In other words, the Trigger Prediction Layer, which is essentially extended to a Mention Prediction Layer, now extracts both entity and event mention spans. Third, the Coreference Prediction Layer computes coreference chains based on the predicted mention spans and their semantic types. Since all the learner sees are mentions, it doesn't know (and doesn't need to know) which coreference chains it computes are entity-based and which ones are event-based. Similarly, it doesn't know (and doesn't need to know) which types in the type inventory are entity types and which ones are event subtypes. A key advantage of this unified model of coreference is that it allows entity and event coreference to be tightly coupled via parameter sharing.

When we apply this model to a test document, we need to distinguish which coreference relations

it identifies are entity-based and which ones are event-based. This can be done easily based on the semantic type associated with the mentions underlying the extracted coreference relation under consideration. If the semantic type is an entity type, the corresponding coreference relation is regarded as an entity coreference relation; otherwise, it is regarded as an event coreference relation.

Argument Prediction Layer To predict arguments and their roles, we pair each top span i and each candidate argument k to form an input vector $\mathbf{va}_{ik} = [\mathbf{g}_i; \mathbf{t}_i; \mathbf{g}_k; \mathbf{t}_k]$, where $\mathbf{g}_i$ is the span representation of i, $\mathbf{t}_i$ is the one-hot subtype vector of i, $\mathbf{g}_k$ is the span representation of argument candidate k, and $\mathbf{t}_k$ is the one-hot subtype vector of k. During training, we use the gold subtype label to derive the subtype vector. During inference, we derive the subtype vector from the output of the Mention Detection Layer. We feed the resulting vector into a FFNN, which outputs a vector $\mathbf{oa}_{ik}$ of dimension 21. $\mathbf{oa}_{ik}(y)$, the yth element of $\mathbf{oa}_{ik}$, is a score indicating k's likelihood of being an argument of i with role y:

$$\mathbf{oa}_{ik} = \text{FFNN}_{oa}(\mathbf{va}_{ik}) \tag{9}$$

$$s_o(i, k, y) = \mathbf{oa}_{ik}(y) \tag{10}$$

Incorporating Consistency Constraints As noted before, we propose to guide the learning process by incorporating commonsense knowledge that encodes *cross-task consistency constraints* on coreference and the auxiliary tasks. We begin by incorporating two consistency constraints on the outputs of coreference and mention detection:
C1: If two spans are coreferent, they should have the same semantic type. **C2**: If a span has an antecedent that is not the dummy antecedent, its semantic type shouldn't be NONE.

We incorporate each constraint into the model via a scoring function that computes how much two spans i (an anaphor) and j (a candidate antecedent of i) should be penalized if a constraint is violated. For constraint **C1**, we define a cost function, c_1, which is computed as follows:

$$c_1(i, j) = \min(|s_t(i, y_i) - s_t(i, y_j)|, |s_t(j, y_j) - s_t(j, y_i)| \tag{11}$$

where $y_i = \arg\max_{y_t} s_t(i, y_t)$ and $y_j = \arg\max_{y_t} s_t(j, y_t)$. Intuitively, c_1 provides an estimate of the least amount of adjustment needed to make i's semantic type the same as j's or the other way round. In particular, c_1 returns 0 (i.e., no penalty) if the two spans have the same type.

Similarly, for constraint **C2**, we define a cost function c_2, which is computed as follows:

$$c_2(i, j) = \begin{cases} 0 & \arg\max_{y \in \mathcal{Y}} s_t(i, y) \neq \text{None} \\ s_t(i, \text{None}) - \max_{y \in \mathcal{Y} \setminus \{\text{None}\}} s_t(i, y) & \text{otherwise} \end{cases} \tag{12}$$

where $\mathcal{Y}$ is the set of possible types. Intuitively, c_2 estimates the minimum amount that needs to be adjusted so that anaphor j's type is not NONE.

Finally, we incorporate c_1 and c_2 into the model as penalty terms in s_c (Equation 1). Specifically, we redefine s_c as follows:

$$s_c(i, j) = \begin{cases} 0 & j = \epsilon \\ s_m(i) + s_m(j) + s_p(i, j) - [\beta_1 c_1(i, j) + \beta_2 c_2(i, j)] & j \neq \epsilon \end{cases} \tag{13}$$

where β_1 and β_2 are positive constants that control the *hardness* of the constraints. The smaller a β_i is, the softer the corresponding constraint is. Intuitively, if a constraint is violated, $s_c(i, j)$ will be lowered by one or more of the penalty terms, and j will less likely be selected as the antecedent of i.

In addition, we enforce the following consistency constraints. Like **C1** and **C2**, each of them will be accompanied by a cost function that will eventually be incorporated into s_c as a penalty term.
Coreference and anaphoricity. **C3**: If a span's antecedent is not the dummy antecedent, its anaphoricity value should be ANAPHORIC. **C4**: If a span has a dummy antecedent, its anaphoricity value should be NON-ANAPHORIC.
Coreference and realis detection. **C5**: If two spans are coreferent, they should have the same realis value. **C6**: If a span's antecedent is not the dummy antecedent, its realis value should not be NONE.
Coreference and argument extraction. **C7**: If two event mention spans are coreferent, their same-role arguments, if any, should be entity-coreferent.

4.2 Training

The loss function we use, $L(\Theta)$, is composed of the losses of the six tasks, and is defined as follows:

$$L(\Theta) = \sum_{i=1}^{d} (\lambda_c L_c + \lambda_t L_t + \lambda_a L_a + \lambda_r L_r + \lambda_o L_o) \tag{14}$$

where the hyperparameters (i.e., the λ's) determine the trade-off between the task losses. The model is trained to minimize $L(\Theta)$, whereas the hyperparameters are tuned using grid search to maximize AVG-F (the standard event coreference evaluation metric; see the next section) on development data.

Task Losses We employ a max-margin loss for each of the six tasks.

Defining the coreference loss is slightly tricky since the coreference annotations for each document are provided in the form of clusters. We adopt the coreference loss function previously defined by Wiseman et al. (2015) for entity coreference resolution. Specifically, let $\text{GOLD}_c(i)$ denote the set of spans preceding span i that are coreferent with i, and y_c^l be $\arg\max_{y \in \text{GOLD}_c(i)} s_c(i, y)$. In other words, y_c^l is the highest scoring (latent) antecedent of i according to s_c among all the antecedents of i. The loss function for coreference is defined as:

$$L_c(\Theta) = \sum_{i=1}^{n} \max_{j \in \mathcal{Y}(i)} (\Delta_c(i,j)(1 + s_c(i,j) - s_c(i, y_c^l)) \quad (15)$$

where $\Delta_c(i, j)$ is a mistake-specific cost function that returns the cost associated with a particular type of error (Durrett and Klein, 2013).[1] Intuitively, the loss function penalizes a span i if the predicted antecedent j has a higher score than the correct latent antecedent y_c^l.

We similarly define the loss for trigger detection:

$$L_t(\Theta) = \sum_{i=1}^{n} \sum_{\hat{l} \neq y_t} \max(0, \Delta_t(i, \hat{l})(1 + s_t(i, \hat{l}) - s_t(i, y_t))) \quad (16)$$

where $\Delta_t(i, \hat{l})$ is a mistake-specific cost function that returns the cost associated with a particular type of error.[1] Intuitively, the loss function penalizes each span for which each of the wrong subtypes $\hat{l}$ has a higher score than the correct subtype y_t according to s_t.

The task losses for anaphoricity determination, realis detection, and argument extraction are all max-margin losses that are defined similarly as the one used for trigger detection.

5 Evaluation

5.1 Experimental Setup

5.1.1 Corpora

We perform training and evaluation on the English corpora used in the TAC KBP 2017 Event Nugget Detection and Coreference task. There are no official training sets: the task organizers simply made available a number of event coreference-annotated corpora for training. We use LDC2015E29, E68, E73, E94, and LDC2016E64 as our training set, which contain 817 documents with 22894 event mentions distributed over 13146 coreference chains[2]. Among these 817 documents, we reserve 82 documents for parameter tuning and use the remaining documents for model training. We report results on the official test set, which consists of 167 documents with 4375 event mentions distributed over 2963 coreference chains.

5.1.2 Evaluation Metrics

Results of event coreference, trigger detection and realis detection are obtained using version 1.8 of the official scorer provided by the KBP 2017 organizers. For event coreference, the scorer employs four scoring metrics, MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF_e (Luo, 2005) and BLANC (Recasens and Hovy, 2011), as well as the unweighted average of their F-scores (AVG-F). Results of trigger detection and realis detection are both expressed in terms of Precision (P), Recall (R) and F-score. The scorer considers (1) a trigger correctly detected if it has an exact match with a gold trigger in terms of boundary and event subtype, and (2) a realis label correctly classified if it has an exact match with a gold trigger in terms of boundary and realis value.

Additionally, we express results of both argument extraction and anaphoricity determination in terms of Precision, Recall and F-score. We consider an event argument correctly extracted if it has an exact match with a gold trigger-argument pair in terms of trigger boundary, event subtype, argument head and argument role. We consider an anaphoric mention correct if it has an exact match with the boundary of a gold anaphoric mention.

Finally, we report entity coreference results in terms of CoNLL score, which is the unweighted average of MUC, B^3, and CEAF_e.

5.1.3 Implementation Details

We use the SpanBERT-large model in the Span Representation Layer.[3] For each document, we split it into segments of length 512 and generate all spans of length up to 10. Each FFNN has one hidden layer of size 2000. The size of the width feature embedding is 20. For span pruning, we keep the top 50% of the spans. For candidate antecedent pruning, we keep the top 15 antecedents.

[1] Space limitations preclude a description of these error types. See Durrett and Klein (2013) for details.

[2] LDC2015E73 and E94 don't have annotations for entity detection, entity coreference resolution and argument extraction. We set the losses of these three tasks to 0 during training.

[3] https://github.com/facebookresearch/SpanBERT

	Event Coreference					Trigger			Anaphoricity			Realis			Argument			Entity Coref.
	MUC	B^3	CEA	BLA	AVG	P	R	F	P	R	F	P	R	F	P	R	F	CoNLL
Jiang et al. (2017)	30.6	43.8	39.9	27.0	35.3	56.8	55.6	56.2	–	–	–	48.0	46.9	47.4	–	–	–	–
Huang et al. (2019)	35.7	43.2	40.0	32.4	36.8	56.8	46.4	51.1	–	–	–	–	–	–	–	–	–	–
Lu and Ng (2020)	37.1	44.5	40.0	29.9	37.9	64.5	46.9	54.3	–	–	–	–	–	–	–	–	–	–
Knowledge-lean	37.6	52.3	51.7	33.6	43.8	71.5	55.3	62.4	–	–	–	–	–	–	–	–	–	–
Pipeline	38.6	53.0	53.0	35.0	44.9	73.9	56.1	63.8	43.0	44.5	43.8	70.0	53.1	60.3	36.9	29.9	33.0	72.6
Full Joint	45.2	54.7	53.8	38.2	48.0	71.6	58.7	64.5	50.4	45.3	47.7	63.7	52.0	57.3	32.4	24.5	27.9	68.7

Table 2: Results of different resolvers on event coreference and related tasks. Results in rows 1-3 are copied verbatim from the original papers; – indicates the corresponding result is not available.

For training, we use document sized mini-batches and apply a dropout rate of 0.3. Following Joshi et al. (2019), we use different learning rates for training the task parameters and the SpanBERT parameters. Specifically, the task learning rate is 1×10^{-5} and is decayed linearly, whereas the learning rate for SpanBERT is 2×10^{-4} and is decayed linearly. The hyperparameters in the loss function, $\lambda_c, \lambda_t, \lambda_a, \lambda_r$, and λ_o, are 1, 1, 0.05, 0.5, and 0.05.

5.2 Results and Discussion

Results are shown in Table 2. To gauge the performance of our model, we employ five baselines. Row 1 shows the results of our first baseline, Jiang et al.'s (2017) resolver, which is the highest-scoring system participating in KBP 2017. Rows 2 and 3 show the performance of our next two baselines, a neural resolver (Huang et al., 2019) and a non-neural resolver (Lu and Ng, 2020) that have achieved the best results to date on the KBP 2017 test set. Hence, these three baselines can be viewed as the prior state of the art. As we can see, while Jiang et al. have the best trigger detector (56.2 F-score), the best event coreference performance is achieved by Lu and Ng's resolver (37.9 AVG-F).

Row 4 shows our fourth baseline, which is our model except that (1) three prediction layers (argument, realis, and anaphoricity) are removed, and (2) the remaining layers are trained to identify event mentions only (i.e., without entity mentions). This baseline mimics typical knowledge-lean approaches to event coreference resolution, which perform only trigger detection and event coreference, but is the first knowledge-lean event coreference approach implemented in a span-based framework. As we can see, this baseline outperforms Lu and Ng's resolver by 5.9% points in AVG-F for event coreference. A closer inspection of the coreference evaluation metrics reveals that in comparison to Lu and Ng, this baseline's B^3, CEAF_e and BLANC scores increase substantially while its MUC score barely changes. Since MUC only rewards successful identification of coreference links, the fact that the MUC score is more or less unchanged implies that the improvement does not arise from link identification; rather, the fact that the B^3, CEAF_e and BLANC scores improve suggests that the improvement arises from successful identification of singleton clusters. This is further supported by the improvement in trigger detection: the baseline's trigger detection module achieves an F-score of 62.4, outperforming Lu and Ng's trigger detection module by 8.1% points in F-score. This huge improvement should not be surprising, as SpanBERT is specifically designed to extract text spans. Overall, despite the encouraging 6%-point improvement in event coreference AVG-F score, we cannot say that the successes of span-based models on entity coreference can be extended to event coreference as it largely fails to establish event coreference links.

Row 5 shows the result of our fifth baseline, which is a pipelined version of our model designed to gauge the benefits of our joint model. Here, we first train a trigger detector, which is the same as the Mention Prediction Layer of our model trained to assign event subtypes to top spans. The resulting triggers are used to train an anaphoricity model (same as our model's Anaphoricity Prediction Layer) and a realis detection model (same as our model's Realis Prediction Layer). Next, we train an entity coreference model, which is the same as our third baseline except that it is trained to operate on entity rather than event mention spans. Then, we train an argument extraction model (same as our model's Argument Prediction Layer) using the extracted entity mentions as candidate arguments for the triggers identified by the trigger detection model. Finally, the outputs of these models are used to enforce the seven constraints in our model as hard constraints: any candidate antecedent of an anaphor that violates any of the constraints is filtered prior to event coreference resolution. Over-

all, this baseline outperforms the fourth baseline by 0.6% points in AVG-F for event coreference and 1.4% points in F-score for trigger detection.

Row 6 shows the result of our full model, which outperforms the Pipeline model by 3.1% points in AVG-F for event coreference and establishes new state-of-the-art results. Encouragingly, the gains in AVG-F are accompanied by improvements w.r.t. all four coreference scoring metrics. In particular, the MUC score improves considerably by 6.6% points, which means that the full model has successfully identified event coreference links. In addition, we see a 0.7% point improvement in trigger detection over Pipeline, and a 12.9% point improvement in realis detection in comparison to Jiang et al. For bookkeeping purposes, we also report the scores for each component of our model. Overall, the fact that our joint model outperforms Pipeline suggests the benefits of joint modeling.

5.3 Model Ablations

To evaluate the contribution of the different components in our model, we show in Table 3 ablation results, which we obtain by removing one component at a time from the model and retraining it.

Consistency constraints. Ablating the consistency constraints means removing all the penalty terms from s_c. The ablated system resembles what one would usually see in a multi-task learning setup, where the different tasks involved has a shared representation. As we can see from row 2, event coreference performance drops by 1% point, suggesting the usefulness of using consistency constraints in a multi-task setup. While it is perhaps not surprising that the consistency constraints have the largest impact on event coreference performance, it is somewhat interesting to see that there is one task whose performance improves when consistency constraints are ablated, realis detection.

Entity coreference. Next, we ablate the entity coreference component. The ablation of entity coreference necessitates the removal of the argument extraction component and the associated constraints since the latter relies on the outputs of entity coreference. We see from row 3 that event coreference performance drops precipitously by 2.7% points. This suggests that entity coreference has a considerable positive impact on event coreference.

The next question is: will coreference performance go up or down if we treat entity and event coreference as two separate tasks that are learned in a typical multi-task setup? As we can see from row 4, the performances of event coreference and entity coreference drop by 0.8% points and 3% points respectively. These results suggest that our viewing the two tasks as a single task is beneficial.

		Event Coref.	**Tri.**	**Ana.**	**Rea.**	**Arg.**	**Entity Coref.**
		AVG	F	F	F	F	CoNLL
1	Full Model	48.0	64.5	47.7	57.3	27.9	68.7
2	− constraints	47.0	64.5	47.6	57.9	27.9	68.5
3	− entity coref.	45.3	63.5	45.0	58.2	–	–
4	sep. entity coref.	47.2	65.1	47.8	56.3	26.0	65.7
5	− anaphoricity	47.5	64.9	46.9	58.1	28.4	69.3
6	− realis	46.6	64.8	46.7	–	29.6	69.3
7	− argument	47.4	64.3	48.6	58.5	–	66.7

Table 3: Ablation results of the full model.

Anaphoricity determination. Next, we ablate the anaphoricity component, which involves removing both its task loss and the associated constraints. From row 5, we see that event coreference performance drops by 0.5% points, and anaphoricity determination performance drops 0.8% points.

Realis detection. When we ablate realis detection, both the task loss and the associated consistency are removed. The performances of event coreference and anaphoricity drop precipitously, by 1.4% points and 1.0% point respectively, suggesting the usefulness of realis detection for both event coreference and anaphoricity detection.

Argument extraction. Finally, when the argument extraction component is ablated, event coreference performance drops by 0.6% points. These results illustrate the importance of argument extraction for event coreference.

Overall, these results suggest that each component contributes positively to event coreference.

6 Conclusion

We proposed the first neural model for event coreference resolution that (1) jointly learned six tasks, (2) used consistency constraints to guide learning, and (3) viewed entity and event coreference as a single task. Our model outperformed several strong baselines and achieved state-of-the-art results on the KBP 2017 event coreference dataset.

Acknowledgments

We thank the three anonymous reviewers for their detailed and insightful comments on an earlier draft of the paper. This work was supported in part by NSF Grants IIS-1528037 and CCF-1848608.

References

Jun Araki, Zhengzhong Liu, Eduard Hovy, and Teruko Mitamura. 2014. Detecting subevent structure for event coreference resolution. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 4553–4558.

Jun Araki and Teruko Mitamura. 2015. Joint event trigger identification and event coreference resolution with structured perceptron. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2074–2080.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the LREC Workshop on Linguistic Coreference*, pages 563–566.

Chen Chen and Vincent Ng. 2016. Joint inference over a lightly supervised information extraction pipeline: Towards event coreference resolution for resource-scarce languages. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2913–2920.

Zheng Chen, Heng Ji, and Robert Haralick. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of the Workshop on Events in Emerging Text Types*, pages 17–22.

Prafulla Kumar Choubey and Ruihong Huang. 2017. Event coreference resolution by iteratively unfolding inter-dependencies among events. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2124–2133.

Prafulla Kumar Choubey and Ruihong Huang. 2018. Improving event coreference resolution by modeling correlations between event coreference chains and document topic structures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 485–495.

Agata Cybulska and Piek Vossen. 2013. Semantic relations between events and their time, locations and participants for event coreference resolution. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 156–163.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982.

Yin Jou Huang, Jing Lu, Sadao Kurohashi, and Vincent Ng. 2019. Improving event coreference resolution by learning argument compatibility from unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 785–795.

Shanshan Jiang, Yihan Li, Tianyi Qin, Qian Meng, and Bin Dong. 2017. SRCB entity discovery and linking (EDL) and event nugget systems for TAC 2017. In *Proceedings of the 2017 Text Analysis Conference*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, (8):64–77.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for coreference resolution: Baselines and analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5803–5808.

Sebastian Krause, Feiyu Xu, Hans Uszkoreit, and Dirk Weissenborn. 2016. Event linking with sentential features from convolutional neural networks. In *Proceedings of the Conference of Computataional Natural Language Learning*, pages 239–249.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 687–692.

Tao Li, Vivek Gupta, Maitrey Mehta, and Vivek Srikumar. 2019. A logic-driven framework for consistency of neural models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3924–3935.

Zhengzhong Liu, Jun Araki, Eduard Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 4539–4544.

Jing Lu and Vincent Ng. 2017. Joint learning for event coreference resolution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 90–101.

Jing Lu and Vincent Ng. 2018. Event coreference resolution: A survey of two decades of research. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 5479–5486.

Jing Lu and Vincent Ng. 2020. Event coreference resolution with non-local information. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 653–663.

Jing Lu, Deepak Venugopal, Vibhav Gogate, and Vincent Ng. 2016. Joint inference for event coreference resolution. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3264–3275.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3036–3046.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Katie McConky, Rakesh Nagi, Moises Sudit, and William Hughes. 2012. Improving event coreference by context extraction and dynamic feature weighting. In *Proceedings of the 2012 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, pages 38–43.

Thien Huu Nguyen, Adam Meyers, and Ralph Grishman. 2016. New York University 2016 system for KBP event nugget: A deep learning approach. In *Proceedings of the 2016 Text Analysis Conference*.

Haoruo Peng, Yangqi Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 392–402.

Marta Recasens and Eduard Hovy. 2011. BLANC: Implementing the Rand Index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference*.

Haoyu Wang, Muhao Chen, Hongming Zhang, and Dan Roth. 2020. Joint constrained learning for event-event relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 696–706.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1416–1426.