Span-Based 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

Motivated by the recent successful application of span-based models to entity-based information extraction tasks, we investigate span-based models for event coreference resolution, focusing on determining (1) whether the successes of span-based models of entity coreference can be extended to event coreference; (2) whether exploiting the dependency between event coreference and the related subtask of trigger detection; and (3) whether automatically computed entity coreference information can benefit span-based event coreference resolution. Empirical results on the standard evaluation dataset provide affirmative answers to all three questions.

1 Introduction

Within-document event coreference resolution is the task of determining which event mentions in a document refer to the same real-world event. Consider the following example:

> Yesterday the Delhi Police $\{\text{slapped}\}_{ev1}$ a protester while she was $\{\text{demonstrating}\}_{ev2}$ outside a hospital. At almost the same time, a woman in her 60s was $\{\text{beaten up}\}_{ev3}$ by policemen in another $\{\text{protest}\}_{ev4}$ in the northern Indian state of Uttar Pradesh. As of now, the Delhi Police has suspended the cop who $\{\text{assaulted}\}_{ev5}$ the woman protester.

In this example, there are five event mentions ($ev1$–$ev5$), which are triggered by the words/phrases *slapped*, *demonstrating*, *beaten up*, *protest*, and *assaulted*, respectively. While $ev1$, $ev3$, and $ev5$ are of subtype ATTACK, only $ev1$ and $ev5$ are coreferent, as $ev3$ took place during a different protest. In addition, $ev2$ and $ev4$ are not coreferent because they refer to different PROTEST events.

Event coreference resolution is arguably more challenging than its entity counterpart, entity coreference resolution. To understand the reason, consider the standard information extraction (IE) pipeline (Figure 1), which involves (1) extracting entity mentions from a given document (the *entity extraction* component) and determining which of them are coreferent (the *entity coreference* component); (2) extracting event mentions by identifying their trigger words/phrases and determining which entity mentions are their arguments/participants (the *event extraction* component); and (3) determining which event mentions are coreferent using information from both the entity pipeline and the event pipeline. As we can see, while an entity coreference resolver has to assume as inputs the noisy outputs of an entity extraction component, an event coreference resolver has to assume as inputs the noisy outputs of a larger set of upstream components in both the entity pipeline and the event pipeline, each of which involves challenging tasks.

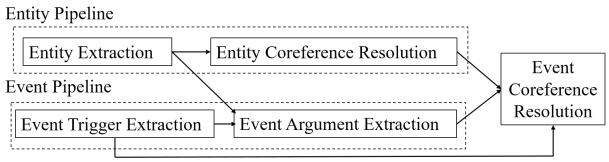


Figure 1: The standard information extraction pipeline.

Despite these challenges, a relatively new type of neural models known as *span-based* models has recently been successfully applied to entity-based IE tasks, including entity extraction and entity coreference resolution (Lee et al. 2017; Luan et al. 2019; Wadden et al. 2019). Span-based models, unlike many other neural models, focus on learning task-specific *span* rather than word representations. For instance, when applied to entity coreference, span-based models seek to learn representations of text spans that correspond to entity mentions so that two coreferent entity mentions have similar span representations. Span-based models have revolutionized the way entity coreference research is conducted: while traditional entity coreference research has focused on designing complex coreference models, span-based models focus on learning span representations that can be used in conjunction with relatively simple coreference models. In addition, while traditionally entity mention extraction is performed prior to entity coreference resolution (see Figure 1), span-based models learn entity mention boundaries as part of the entity coreference resolution process. This mitigates the propagation of errors from the entity extraction component to the entity coreference resolution component.

Our work is motivated by our belief that the successful application of span-based models to entity-based IE tasks creates new opportunities for event coreference research. In particular, our work is driven by three research questions.

First, *how well would span-based models work for event*

coreference resolution? As mentioned above, event coreference is more challenging than its entity counterpart. Structurally, an event mention is more complex than an entity mention: while an entity mention is composed of the mention span and its entity type, an event mention is composed of not only the *trigger* but also the *arguments* (i.e., the event participants) and the *roles* they play in the event. Determining whether two event mentions are coreferent requires not only that their event subtypes are the same, but also that their corresponding arguments are entity-coreferent. In other words, unlike entity coreference where much of the information needed to determine coreference is encoded within the entity mention itself, event coreference depends heavily on the context in which an event trigger appears. Whether span-based models can effectively learn representations of event mentions based purely on knowledge provided by contextualized embeddings is an interesting question.

Second, *what would be the best way to exploit the dependency between event coreference and trigger detection in span-based event coreference models?* Trigger detection is the subtask of event coreference that involves identifying the event mentions in a text document and determining their event subtypes. While span-based models enable trigger detection and event coreference to be learned simultaneously in a multi-task learning framework, they fail to exploit an important cross-task consistency constraint between trigger detection and event coreference: two event mentions with different event subtypes cannot be coreferent. In other words, it is possible for a span-based model to posit two event mentions having different event subtypes as coreferent. Motivated by this potential weakness, we explore different methods for exploiting this cross-task dependency.

Third, *can entity coreference information benefit event coreference resolution?* Intuitively, the answer is yes. Recall from our running example that $ev3$ is not coreferent with $ev1$ and $ev5$ and that $ev2$ and $ev4$ are not coreferent. Determining such non-coreference relations requires knowledge of *entity coreference* over their arguments/participants. For instance, while the victim in each of $ev1$, $ev3$ and $ev5$ was a woman, only $ev1$ and $ev5$ are coreferent since their attackers, the cop in the Delhi Police, are entity-coreferent. In contrast, $ev3$ is not coreferent with $ev1$ and $ev5$ because its attackers, the policemen in Uttar Pradesh, are not entity-coreferent with the Delhi Police. Despite the usefulness of entity coreference information for event coreference, the vast majority of existing event coreference resolvers do *not* use entity coreference information because of the noise inherent in computing it. Lee et al.'s (2012) work represents one of the few attempts that employ entity coreference information for event coreference and show that entity coreference information can be profitably used for event coreference resolution. Nevertheless, we believe that it is time to revisit Lee et al.'s results. The reason is that the event coreference model they used (i.e., the so-called *mention-pair* model (Soon, Ng, and Lim 2001; Ng and Cardie 2002)) is considered a weak model in today's standard. In other words, while they showed that entity coreference information can be used to improve a *weak* event coreference model, it is not clear whether it can improve *strong* event coreference models.

In sum, our contributions in this paper are two-fold. First, we investigate the application of span-based models to event coreference resolution and the exploitation of cross-task consistency constraints and entity coreference information in span-based event coreference resolution. Second, results on the KBP 2017 event coreference dataset demonstrate the effectiveness of our span-based event coreference models, especially when augmented with consistency constraints and entity coreference information. In particular, the resulting model achieves state-of-the-art results on this dataset.

2 Related Work

Event coreference: models and features. While there exist heuristic models (e.g., Lu and Ng (2016)), unsupervised models (e.g., Bejan and Harabagiu (2014), Chen and Ng (2015)) and semi-supervised models (e.g., Peng, Song, and Roth (2016)) for event coreference resolution, the vast majority of event coreference resolvers are supervised (e.g., Ahn (2006), Chen, Ji, and Haralick (2009), Nguyen, Meyers, and Grishman (2016)). Broadly, supervised approaches can be divided into two categories. In *pipeline* models (e.g., Choubey and Huang (2017), Choubey and Huang (2018)), event mentions are first extracted in the trigger detection component and then used for event coreference. While most of the existing approaches to event coreference are pipeline-based, errors in trigger detection can propagate to the event coreference component. To address error propagation, researchers developed *joint* models, including joint inference models using Integer Linear Programming (Chen and Ng 2016) and Markov Logic Networks (Lu et al. 2016), as well as joint learning models using structured perceptrons (Araki and Mitamura 2015) and structured conditional random fields (CRFs) (Lu and Ng 2017). Our span-based model is the first joint event coreference model that is neural-based.

There is a large body of work on designing features for supervised event coreference resolution. These features can broadly be divided into four categories: lexical (e.g., features that encode string-matching facilities) (Lee et al. 2012), semantic (e.g., features that encode semantic similarity measures) (Liu et al. 2014), argument-based (i.e., features that determine how compatible the arguments of two event mentions are) (Yang, Cardie, and Frazier 2015), and discourse-based (e.g., the token and sentence distances between two event mentions) (Cybulska and Vossen 2015). In contrast, span-based models operate without feature engineering.

Exploiting cross-task constraints for event coreference. The cross-task constraint that two coreferent event mentions must have the same event subtype has been exploited in both pipeline and joint models of event coreference. In pipeline approaches, the subtypes predicted by the trigger detector for the two event mentions under consideration have been used as features and as constraints for event coreference. Specifically, some coreference models have been trained on a feature that encodes whether two event mentions have the same subtype (Chen and Ng 2014), while others have used the subtypes as a *hard* constraint to disallow two event mentions with different subtypes to be posited as coreferent during resolution (Chen and Ji 2009). In Lu and Ng's (2017)

structured CRF-based joint event coreference model, these cross-task consistency constraints are implemented as *soft* constraints in the form of binary and ternary factors.

As noted before, we will explore different methods for exploiting the dependency between trigger detection and event coreference in span-based event coreference models. Regardless of whether we encode this dependency as a feature or as a hard/soft constraint, we believe that the impact of explicitly encoding and exploiting this dependency on span-based coreference models is likely to be greater than that on traditional models. The reason is that this dependency will be exploited to *train* our span-based models, meaning that it will have an impact on the learned span representations.

Using entity coreference information for event coreference. Lee et al. (2012) employ entity coreference information for event coreference. Using heuristically extracted seeds consisting of pairs of mentions that are either entity- or event-coreferent, they iteratively bootstrap event coreference output using entity coreference output and vice versa, and show that event coreference performance can be improved using entity coreference information. Nevertheless, their resolver is considered a fairly weak model given today's technologies: it has since been superseded by models that no longer make locally optimal decisions involving only two (clusters of) event mentions. In particular, it is not clear whether entity coreference information can improve *strong* event coreference models, such as the span-based event coreference model that we will describe in the next section.

For a comprehensive survey of recent related work on event coreference, we refer the reader to Lu and Ng (2018).

3 Baseline Event Coreference Model

To determine how well span-based models work for event coreference resolution, we begin by designing a span-based event coreference model, which we will use as a baseline and augment with cross-task constraints and entity coreference information in subsequent sections. This 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. Each such span corresponds to a candidate trigger, and the model simultaneously learns trigger detection and event coreference resolution, which we define below.

The trigger detection task aims to assign each span i a subtype 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 the subtype of i 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 each y_c is the id of its 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 i's likelihood of being a trigger and $s_p(i,j)$ is a pairwise coreference score computed over 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 $\mathcal{Y}(i)$, the set of candidate antecedents of i, contains all spans preceding i in the associated document.

Model Structure

The model structure (Figure 2(a)) is described 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 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:

$$s_m(i) = \mathbf{w}_m \cdot \mathrm{FFNN}_m(\mathbf{g}_i) \quad (2)$$

where FFNN is a 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:

$$\mathbf{ot}_i = \mathrm{FFNN}_t(\mathbf{g}_i) \quad (3)$$

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

Coreference Prediction Layer. To predict event coreference links, we first calculate the pairwise score between spans i and j as follows:

$$s_p(i,j) = \mathbf{w}_c \cdot \mathrm{FFNN}_c([\mathbf{g}_i; \mathbf{g}_j; \mathbf{g}_i \circ \mathbf{g}_j, \mathbf{u}_{ij}]) \quad (5)$$

where $\circ$ denotes element-wise multiplication, $\mathbf{g}_i \circ \mathbf{g}_j$ encodes the similarity between span i and span j, and $\mathbf{u}_{ij}$ is a feature embedding encoding the distance between two spans. We can then compute the coreference score defined in Equation 1 using Equations 2 and 5.

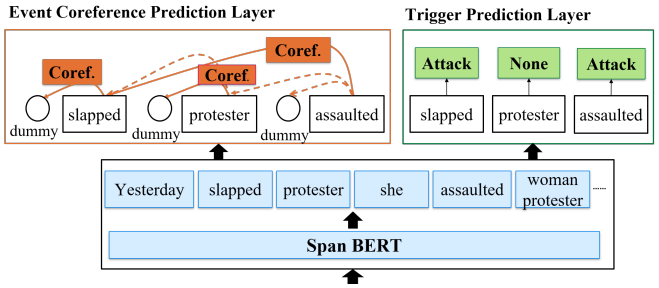
To improve running time, we follow Lee, He, and Zettlemoyer (2018) and use their antecedent pruning method.

Training

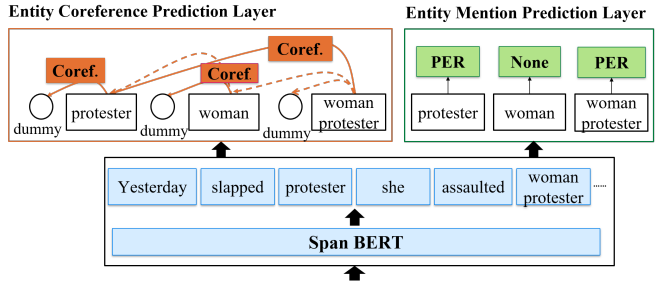
The loss function we use, $L(\Theta)$, is composed of the losses of two tasks, and is defined as follows:

$$L(\Theta) = \sum_{i=1}^{d} (\lambda_c L_c + \lambda_t L_t) \quad (6)$$

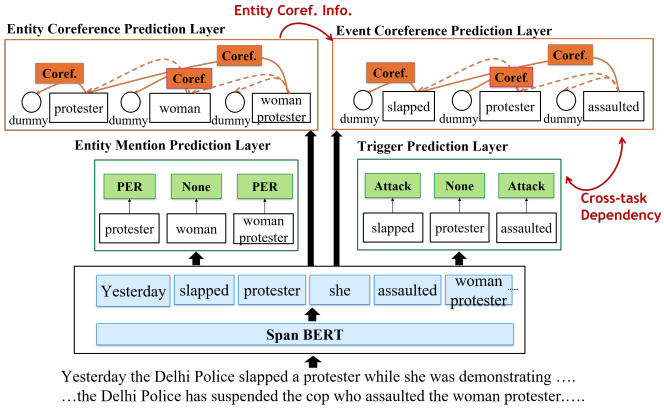
where hyperparameters λ_t and λ_c determine the trade-off between two task losses. The model is trained to minimize $L(\Theta)$, whereas the hyperparameters are tuned using grid search to maximize AVG-F (the standard event coreference evaluation metric; see Section 6) on development data.



(a) Baseline event coreference model



(b) Independent entity coreference model



(c) Joint model

Figure 2: Model architectures.

Task Losses. We employ a max-margin loss for each task.

Defining the coreference loss is tricky since the coreference annotations for each document are provided in the form of clusters. We adopt the coreference loss function 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 coreference loss function is defined as:

$$L_c(\Theta) = \sum_{i=1}^{n} \max_{j \in \mathcal{Y}(i)} (\Delta_c(i,j)(1 + s_c(i,j) - s_c(i, y_c^l)) \quad (7)$$

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:

$$L_t(\Theta) = \sum_{i=1}^{n} \sum_{\hat{l} \neq y_t} \max(0, \Delta_t(i, \hat{l})(1 + s_t(i, \hat{l}) - s_t(i, y_t))) \quad (8)$$

where $\Delta_t(i, \hat{l})$ is a mistake-specific cost function that returns the cost associated with a particular type of error.[2] 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.

4 Exploiting Cross-Task Dependency

While the multi-task learning setup employed by the Baseline allows trigger detection and event coreference to benefit each other via the shared representation layer, it fails to exploit the dependency that exist between them. Below we examine four methods for exploiting this dependency.

Gold feature (GF). In the first method, we train the Baseline coreference model with an additional feature that encodes whether the two event mentions under consideration have the same subtype. During training, we compute this feature using *gold* subtypes. During testing, we compute this feature using the subtypes predicted by the trigger detector.

Predicted feature (PF). Our second method is essentially the same as the first method, except that the feature is computed using the event subtypes predicted by the trigger detector during *both* training and testing. This allows the model to be trained in the same setup in which it will be evaluated.

Hard constraint (HC). In the first two methods, the dependency between trigger detection and event coreference is encoded *implicitly* as a feature, so it is possible for the event coreference model to posit two event mentions having different subtypes as coreferent. Our third method encodes this dependency explicitly as a *hard* constraint. Specifically, given an event mention to be resolved, we filter all of its candidate antecedents whose *gold* subtypes are different from that of the event mention during training. During testing, we filter based on the predicted subtypes.

Soft constraint (SC). Finally, we explore a new method for encoding this dependency as *soft* consistency constraints. Specifically, we incorporate into the Baseline the following two consistency constraints on the outputs of trigger detection and event coreference.

$\mathbf{P}_1$: If two spans do not have the same event subtype, they

[1] There are three error types: (1) false link (incorrectly resolved anaphoric mentions); (2) false new (anaphoric mentions misclassified as non-anaphoric); and (3) wrong link (non-anaphoric mentions misclassified as anaphoric). We use hyperparameters α_{c1}, α_{c2}, α_{c3} to adjust their trade-offs.

[2] We define three error types: (1) non-triggers misclassified as triggers, (2) triggers misclassified as non-triggers, and (3) triggers labeled with the wrong subtype. We use hyperparameters α_{t1}, α_{t2}, α_{t3} to determine their trade-offs.

cannot be coreferent.

$\mathbf{P}_2$: If a span has NONE as its event subtype, its antecedent must be the dummy antecedent.

To make use of each constraint in the event coreference model, we define a penalty function, which imposes a penalty on two spans i and j if they violate a constraint, where i is an anaphor and j is a candidate antecedent of i. For $\mathbf{P}_1$, the penalty function p_1 is computed as follows:

$$p_1(i,j) = \min(|s_t(i,y_i) - s_t(i,y_j)|, |s_t(j,y_j) - s_t(j,y_i)|) \quad (9)$$

where $y_i = \arg\max_{y_t} s_t(i,y_t)$ and $y_j = \arg\max_{y_t} s_t(j,y_t)$. Informally, p_1 returns the minimum amount of adjustment needed to ensure that i and j have the same event subtype.

For $\mathbf{P}_2$, we employ the following penalty function p_2:

$$p_2(i) = \begin{cases} 0 & \arg\max_{y \in \mathcal{Y}} s_t(i,y) \neq \text{None} \\ s_t(i,\text{None}) - \max_{y \in \mathcal{Y} \setminus \{\text{None}\}} s_t(i,y) & \text{otherwise} \end{cases} \quad (10)$$

where $\mathcal{Y}$ is the set of possible subtypes. Informally, p_2 returns the smallest amount of adjustment needed to ensure that the subtype of i is not NONE.

Finally, we employ these two penalty functions in the coreference model by updating s_c (Equation 1) as follows:

$$s_c(i,j) = \begin{cases} 0 & j = \epsilon \\ s_m(i) + s_m(j) + s_p(i,j) - [\gamma_1 p_1(i,j) + \gamma_2 p_2(i)] & j \neq \epsilon \end{cases} \quad (11)$$

where γ_1 and γ_2 are positive constants. To see why we incorporate p_1 and p_2 into s_c, recall that p_k will return a positive value if span i and span j violate constraint P_k. This will in turn decrease the value of $s_c(i,j)$, thus making it less likely for j to be chosen as the antecedent of i. Note that γ_1 and γ_2 are the weights associated with p_1 and p_2. For instance, setting γ_k to a smaller value will undermine the effect of p_k and thus soften the corresponding constraint P_k.

5 Exploiting Entity Coreference Information

In this section, we first describe how we compute entity coreference chains and then show how we exploit the resulting chains for span-based event coreference resolution.

Computing Entity Coreference Chains

To compute entity coreference chains, we use two models.

Independent model. The first model is an *independent* model, which is trained to compute coreference chains over the entity mentions in a given document independently of the Baseline event coreference model.

The structure of the Independent model, which is shown in Figure 2(b), mirrors that of the event coreference model. Specifically, the entity coreference model also has three layers: (1) the span representation layer, which learns the spans corresponding to entity mentions; (2) the entity mention prediction layer, which assigns to each span an entity type taken from a repository of predefined entity types or NONE if the span does not correspond to an entity mention; and (3) the entity coreference layer, which computes coreference links over spans augmented with its predicted entity type. The loss function is a weighted combination of two losses, one corresponding to entity mention detection, which predicts entity mentions and their entity types, and the other corresponding to entity coreference, whose loss is defined in a similar manner as the one used for event coreference.

Joint model. One weakness of the Independent model is that the interaction between entity coreference and event coreference is minimal. To more tightly couple the two tasks, we train the Independent entity coreference model and the Baseline event coreference model *jointly*.

The structure of the Joint model is shown in Figure 2(c). When trained in a joint fashion, Independent and Baseline share the span representation layer, and the loss function employed by Joint is a weighted combination of the loss functions employed by Independent and Baseline.

Incorporating Entity Coreference Information

We explore two methods for exploiting entity coreference information for span-based event coreference resolution.

As a hard constraint. Our first method uses the entity coreference information to create a constraint between entity coreference and event coreference, which specifies that two event mentions cannot be coreferent if their corresponding arguments are not entity-coreferent. For instance, recall that the ATTACKER arguments $ev1$ and $ev3$ in our running example are not entity-coreferent, so this constraint will be violated if $ev1$ and $ev3$ are predicted to be event-coreferent. We implement this constraint as a *hard* constraint, enforcing it during both training and testing by pruning any candidate antecedent that violates this constraint with its anaphor.

Implementing this constraint requires that we (1) identify the arguments of an event mention and (2) compute their semantic roles. For simplicity, we approximate these two tasks as follows: we consider an entity mention en as an argument of an event mention ev if en is reachable from ev's trigger via a dependency path of length less than 4; moreover, we set en's "semantic role" w.r.t. ev to be the sequence of dependency labels over the dependency path connecting them. Dependency relations are obtained using the Stanford CoreNLP toolkit (Manning et al. 2014).

As features. Our second method uses the entity coreference information to compute three binary features for training the event coreference model. The first feature encodes whether the two event mentions under consideration have any semantic role in common. The second feature encodes whether any of their arguments having the same semantic role are not entity-coreferent. The third feature is complementary to the second feature: its value is 1 if and only if the value of the second feature is 0.

These two methods for exploiting entity coreference information for span-based event coreference resolution can be used in combination with any of the two aforementioned models for computing event coreference chains. This results in four combinations that we will evaluate in the next section, namely **Independent/Feature (IF)**, **Independent/Constraint (IC)**, **Joint/Feature (JF)**, and **Joint/Constraint (JC)**.

6 Evaluation

Experimental Setup

Datasets. We employ the English corpora made available to us as part of the TAC KBP 2017 Event Nugget Detection and Coreference task. For training, we use LDC2015E29, E68, E73, E94 and LDC2016E72. For testing, we use the official KBP 2017 evaluation set (Mitamura, Liu, and Hovy 2017). Statistics on these datasets are shown in Table 1.

Evaluation metrics. Results of event coreference are obtained using version 1.8 of the official scorer provided by the KBP 2017 shared task organizers. This scorer reports results in terms of AVG-F, which is the unweighted average of the F-scores of four coreference evaluation metrics, namely MUC (Vilain et al. 1995), B^3 (Bagga and Baldwin 1998), CEAF_e (Luo 2005) and BLANC (Recasens and Hovy 2011).

For completeness we also report the results of trigger detection, entity coreference, and entity detection. Trigger detection results are expressed in terms of recall (R), precision (P), and F-score (F), where a trigger is considered correctly detected if it has an exact match with a gold trigger in terms of boundary and event subtype. For entity coreference, we express results in terms of the CoNLL score (Pradhan et al. 2012), which is the unweighted average of the MUC, B^3, and CEAF_e F-scores. Entity detection results are expressed in terms of F-score, where an entity mention is considered correctly detected if it has an exact match with a gold entity mention in terms of boundary and type.

Implementation details. We use SpanBERT-large in the span representation layer.[3] We split each document into segments of length 512 and generate all spans of length up to 10. Each FFNN has 1 hidden layer of size 3000. The size of the width feature embedding is 20. For span pruning, we keep the top 30% of the spans. For candidate antecedent pruning, we keep the top 20 antecedents. For training, we use document sized mini-batches. We 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. For the hyperparameters in the loss function, we search for λ_i, the weight of each task, out of $\{1, 5, 10\}$. For the weights associated with the task errors, we search out of $\{0.1, 0.5, 1, 5, 10, 20, 50, 100\}$.

Results and Discussion

We report results of experiments that can provide answers to the three research questions mentioned in the introduction.

Research question 1. Recall that the first question concerns how well span-based models work for event coreference resolution. To answer this question, we compare our Baseline model (see Section 2) with two state-of-the-art event coreference models. The first one is Huang et al.'s (2019) resolver, which is a neural (but not span-based) resolver that leverages the argument compatibility information acquired from a large unlabeled corpus. The second one is Lu and Ng's (2020) resolver, which extends Lu and Ng's (2017) structured CRF-based resolver by incorporating topic and discourse information. As we can see from Table 2, our Baseline model outperforms the better state-of-the-art model by 5.9% points in event coreference AVG-F score and 8.1% points in trigger detection F-score. For trigger detection, the increase in F-score is accompanied by large gains in both recall and precision. To understand whether the substantial improvement in event coreference stems from better identification of coreference links or better identification of singleton clusters, we also report the MUC F-score in Table 2. Given the modest improvement of the Baseline's MUC F-score over that of Lu and Ng, we can conclude that the improvement in AVG-F stems largely from better identification of singleton clusters. Overall, our Baseline model achieves state-of-the-art results on the KBP 2017 test set.

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

	Train	Dev	Test
#docs	735	82	167
#event mentions	20458	2436	4375
#event chains	12988	1806	2963
#entity mentions	43450	8161	13860
#entity chains	15094	3180	5482

Table 1: Dataset statistics.

	Coreference		Trigger Detection		
System	MUC	AVG F	P	R	F
Huang et al. (2019)	35.7	36.8	56.8	46.4	51.1
Lu and Ng (2020)	37.1	37.9	64.5	46.9	54.3
Baseline	**37.6**	**43.8**	**71.5**	**55.3**	**62.4**

Table 2: Comparison with the current state of the art.

	Coreference		Trigger Detection		
Method	MUC	AVG-F	P	R	F
BL	37.6	43.8	71.5	**55.3**	62.4
GF	40.5	**44.6**	73.6	54.8	**62.8**
PF	40.1	44.1	**74.1**	53.5	62.1
HC	38.9	43.9	73.4	54.2	62.4
SC	**40.9**	44.5	73.9	53.3	62.0

Table 3: Results of using different methods for exploiting the dependency between trigger detection and coreference.

Research question 2. Next, we seek to understand whether exploiting the dependency between event coreference and trigger detection can improve coreference performance by applying the four methods described in Section 4 to the Baseline model.

Results are shown in Table 3. For comparison purposes, we show the results of the Baseline (BL), which does not exploit cross-task dependency, in row 1. For trigger detection, the four methods consistently yield higher precision scores and lower recall scores in comparison to BL; moreover, except for GF, their F-scores are not better than that of BL. Nevertheless, in terms of coreference AVG-F, the four methods consistently outperform BL. Note that the gains in

	Event										Entity					
	Coreference (AVG-F)					Trigger Detection (F)					Coreference (CoNLL)			Mention Detection (F)		
	NC	IF	IC	JF	JC	NC	IF	IC	JF	JC	IF/IC	JF	JC	IF/IC	JF	JC
BL	43.8	44.9	45.1	46.4	46.7	62.4	63.1	64.2	64.1	64.6	**71.2**	64.9	62.2	**86.8**	81.7	80.0
GF	44.6	45.9	45.9	47.0	46.7	62.8	62.8	63.6	65.0	65.1	**71.2**	62.8	62.0	**86.8**	79.3	80.8
PF	44.1	45.3	45.9	46.6	46.5	62.1	63.7	63.5	65.3	64.3	**71.2**	61.4	62.8	**86.8**	78.7	80.8
HC	43.9	45.3	45.7	46.6	46.6	62.4	62.9	64.0	64.1	64.9	**71.2**	65.1	54.8	**86.8**	82.1	71.1
SC	44.5	46.1	45.8	**47.1**	47.0	62.0	63.8	64.3	65.1	**65.7**	**71.2**	60.9	54.8	**86.8**	78.6	71.0

Table 4: Results of exploiting cross-task dependency and entity coreference information on four tasks.

AVG-F are always accompanied by gains in MUC F-score, meaning that the four methods allow additional coreference links to be discovered. Overall, these results suggest that exploiting the dependency between event coreference and trigger detection can indeed improve coreference performance, with GF and SC offering the largest improvements.

Research question 3. Finally, we examine whether entity coreference information can benefit event coreference by evaluating the four combinations for computing and exploiting entity coreference information described in Section 5.

Results of the four tasks (namely, event coreference, trigger detection, entity coreference, and entity mention detection) are shown in Table 4. For each task, columns 2-5 show the four combinations of exploiting entity coreference information. For comparison purposes, we show in the "NC" columns the results obtained *without* entity coreference information. The rows correspond to the different methods for exploiting the cross-task dependency between event coreference and trigger detection, and can be interpreted in the same manner as those in Table 3. Each number in Table 4 therefore corresponds to a specific combination of method for exploiting cross-task dependency and method for exploiting entity coreference chains. Note that the BL/NC combination corresponds to the Baseline described in Section 2.

Consider first the results of event coreference and trigger detection. As we can see, stronger event coreference results are usually accompanied by better trigger detection results. This is perhaps not surprising, as an event coreference link will not be considered correct unless the underlying triggers also have the correct subtype. In addition, using entity coreference information consistently and considerably improves trigger detection and event coreference performance regardless of which method for exploiting cross-task dependency is used. In particular, the Joint results (JF and JC) are better than the Independent results (IF and IC), which in turn are better than those when entity coreference information is not used. These results suggest that tightly coupling the two coreference tasks can improve event coreference performance. Similar to the results in Table 3, the best event coreference results are obtained using SC.

A closer examination of the system outputs reveals that exploiting entity coreference information facilitates the identification of event triggers that are not seen in the training set and the coreference link between them in those sentences that contain both entity and event coreference links. For example, in the sentence "President Vladimir Putin sent his condolences to U.S. President Barack Obama on Tuesday over the deadly tornado that struck Oklahoma City. In his cable, Putin expressed his compassion for relatives of the victims of the powerful twister", the Joint and Independent models successfully posit the event mentions "condolences" and "cable" as coreferent, while the Baseline fails to do so. The reason is that the occurrences of "his" in the two sentences are predicted as coreferent, which in turn helps predict "condolences" and "cable" as coreferent.

Next, consider the results of entity coreference and entity mention detection. The IF/IC results are identical across different methods for exploiting cross-task dependency. Recall that the Independent entity coreference model is trained independently of the event coreference model. Hence, the entity coreference and mention detection results no longer depend on how cross-task dependency and entity coreference information are exploited for event coreference resolution.

Interestingly, the differences between the Independent model results (IF and IC) and the Joint model results (JF and JC) are much larger for entity mention detection than for trigger detection. A closer examination of the system outputs reveals that compared to the Independent model, the Joint model achieves poorer F-scores in mention detection owing to large gains in precision accompanied by even larger drops in recall. This is somewhat surprising, since joint models are typically expected to outperform their independent counterparts. We speculate that the poorer results achieved by the Joint model is the consequence of parameter tuning: since the parameters are tuned to maximize event coreference AVG-F on development data, Joint's solid event coreference performance may have been achieved at the expense of its entity coreference performance. Additional experiments are needed to determine the reason, however.

The drastic performance difference between the Independent model and the Joint model for mention detection is also reflected in the entity coreference results. This should not be surprising, as entity coreference performance depends heavily on successful identification of entity mentions.

7 Conclusion

We explored span-based neural event coreference resolution, showing that our adaptation of Joshi et al.'s (2019) span-based entity coreference model to event coreference resolution already resulted in a baseline model that outperformed the previous state of the art on the KBP 2017 event coreference dataset. We further demonstrated that cross-task dependency and entity coreference information could be profitably exploited for span-based event coreference resolution.

Acknowledgments

We thank the three anonymous reviewers for their comments on an earlier draft of the paper. This work was supported in part by NSF Grants IIS-1528037 and CCF-1848608. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies of the NSF.

References

Ahn, D. 2006. The Stages of Event Extraction. In *Proceedings of the COLING/ACL Workshop on Annotating and Reasoning about Time and Events*, 1–8.

Araki, J.; and Mitamura, T. 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*, 2074–2080.

Bagga, A.; and Baldwin, B. 1998. Algorithms for Scoring Coreference Chains. In *Proceedings of the LREC Workshop on Linguistic Coreference*, 563–566.

Bejan, C.; and Harabagiu, S. 2014. Unsupervised Event Coreference Resolution. *Computational Linguistics* 40(2): 311–347.

Chen, C.; and Ng, V. 2014. SinoCoreferencer: An End-to-End Chinese Event Coreference Resolver. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, 4532–4538.

Chen, C.; and Ng, V. 2015. Chinese Event Coreference Resolution : An Unsupervised Probabilistic Model Rivaling Supervised Resolvers. In *Proceedings of Human Language Technologies: The 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, 1097–1107.

Chen, C.; and Ng, V. 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*, 2913–2920.

Chen, Z.; and Ji, H. 2009. Graph-based Event Coreference Resolution. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, 54–57.

Chen, Z.; Ji, H.; and Haralick, R. 2009. A Pairwise Event Coreference Model, Feature Impact and Evaluation for Event Coreference Resolution. In *Proceedings of the International Workshop on Events in Emerging Text Types*, 17–22.

Choubey, P. K.; and Huang, R. 2017. Event Coreference Resolution by Iteratively Unfolding Inter-dependencies among Events. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2124–2133.

Choubey, P. K.; and Huang, R. 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, 485–495.

Cybulska, A.; and Vossen, P. 2015. Translating Granularity of Event Slots into Features for Event Coreference Resolution. In *Proceedings of the 3rd Workshop on EVENTS*, 1–10.

Durrett, G.; and Klein, D. 2013. Easy Victories and Uphill Battles in Coreference Resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1971–1982.

Huang, Y. J.; Lu, J.; Kurohashi, S.; and Ng, V. 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*, 785–795.

Joshi, M.; Chen, D.; Liu, Y.; Weld, D. S.; Zettlemoyer, L.; and Levy, O. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics* (8): 64–77.

Joshi, M.; Levy, O.; Zettlemoyer, L.; and Weld, D. 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*, 5803–5808.

Lee, H.; Recasens, M.; Chang, A.; Surdeanu, M.; and Jurafsky, D. 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*, 489–500.

Lee, K.; He, L.; Lewis, M.; and Zettlemoyer, L. 2017. End-to-end Neural Coreference Resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 188–197.

Lee, K.; He, L.; and Zettlemoyer, L. 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)*, 687–692.

Liu, Z.; Araki, J.; Hovy, E.; and Mitamura, T. 2014. Supervised Within-Document Event Coreference using Information Propagation. In *Proceedings of the 9th International Conference on Language Resources and Evaluation Conference*, 4539–4544.

Lu, J.; and Ng, V. 2016. Event Coreference Resolution with Multi-Pass Sieves. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, 3996–4003.

Lu, J.; and Ng, V. 2017. Joint Learning for Event Coreference Resolution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 90–101.

Lu, J.; and Ng, V. 2018. Event Coreference Resolution: A Survey of Two Decades of Research. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 5479–5486.

Lu, J.; and Ng, V. 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*, 653–663.

Lu, J.; Venugopal, D.; Gogate, V.; and Ng, V. 2016. Joint Inference for Event Coreference Resolution. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, 3264–3275.

Luan, Y.; Wadden, D.; He, L.; Shah, A.; Ostendorf, M.; and Hajishirzi, H. 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*, 3036–3046.

Luo, X. 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*, 25–32.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J.; and McClosky, D. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60.

Mitamura, T.; Liu, Z.; and Hovy, E. 2017. Events Detection, Coreference and Sequencing: What's next? Overview of the TAC KBP 2017 Event Track. In *Proceedings of the 2017 Text Analysis Conference*.

Ng, V.; and Cardie, C. 2002. Improving Machine Learning Approaches to Coreference Resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 104–111.

Nguyen, T. H.; Meyers, A.; and Grishman, R. 2016. New York University 2016 System for KBP Event Nugget: A Deep Learning Approach. In *Proceedings of the 2016 Text Analysis Conference*.

Peng, H.; Song, Y.; and Roth, D. 2016. Event Detection and Co-reference with Minimal Supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 392–402.

Pradhan, S.; Moschitti, A.; Xue, N.; Uryupina, O.; and Zhang, Y. 2012. CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, 1–40.

Recasens, M.; and Hovy, E. 2011. BLANC: Implementing the Rand Index for Coreference Evaluation. *Natural Language Engineering* 17(4): 485–510.

Soon, W. M.; Ng, H. T.; and Lim, D. C. Y. 2001. A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics* 27(4): 521–544.

Vilain, M.; Burger, J.; Aberdeen, J.; Connolly, D.; and Hirschman, L. 1995. A Model-Theoretic Coreference Scoring Scheme. In *Proceedings of the Sixth Message Understanding Conference*.

Wadden, D.; Wennberg, U.; Luan, Y.; and Hajishirzi, H. 2019. Entity, Relation, and Event Extraction with Contextualized Span Representations. In *Proceedings of Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 5783–5788.

Wiseman, S.; Rush, A. M.; Shieber, S. M.; and Weston, J. 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)*, 1416–1426.

Yang, B.; Cardie, C.; and Frazier, P. 2015. A Hierarchical Distance-dependent Bayesian Model for Event Coreference Resolution. *Transactions of the Association for Computational Linguistics* 3: 517–528.