A Context-Dependent Gated Module for Incorporating Symbolic Semantics into Event Coreference Resolution

Tuan Lai, Heng Ji
University of Illinois at Urbana-Champaign
{tuanml2, hengji}@illinois.edu
Trung Bui, Quan Hung Tran, Franck Dernoncourt, Walter Chang
Adobe Research
{bui, qtran, franck.dernoncourt, wachang}@adobe.com

Abstract

Event coreference resolution is an important research problem with many applications. Despite the recent remarkable success of pre-trained language models, we argue that it is still highly beneficial to utilize symbolic features for the task. However, as the input for coreference resolution typically comes from upstream components in the information extraction pipeline, the automatically extracted symbolic features can be noisy and contain errors. Also, depending on the specific context, some features can be more informative than others. Motivated by these observations, we propose a novel context-dependent gated module to adaptively control the information flows from the input symbolic features. Combined with a simple noisy training method, our best models achieve state-of-the-art results on two datasets: ACE 2005 and KBP 2016.[1]

1 Introduction

Within-document event coreference resolution is the task of clustering event mentions in a text that refer to the same real-world events (Lu and Ng, 2018). It is an important research problem, with many applications (Vanderwende et al., 2004; Ji and Grishman, 2011; Choubey et al., 2018). Since the trigger of an event mention is typically the word or phrase that most clearly describes the event, virtually all previous approaches employ features related to event triggers in one form or another. To achieve better performance, many methods also need to use a variety of additional symbolic features such as event types, attributes, and arguments (Chen et al., 2009; Chen and Ji, 2009; Zhang et al., 2015; Sammons et al., 2015; Lu and Ng, 2016; Chen and Ng, 2016; Duncan et al., 2017). Previous neural methods (Nguyen et al., 2016; Choubey and Huang, 2017; Huang et al., 2019) also use non-contextual word embeddings such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014).

... we are seeing these soldiers {head out}$_{ev1}$ ...
... these soldiers **were set to** {leave}$_{ev2}$ in January ...
ev1 (`Movement:Transport`): Modality = ASSERTED
ev2 (`Movement:Transport`): Modality = OTHER

Table 1: An example of using the modality attribute to improve event coreference resolution.

With the recent remarkable success of language models such as BERT (Devlin et al., 2019) and SpanBERT (Joshi et al., 2020), one natural question is whether we can simply use these models for coreference resolution without relying on any additional features. We argue that it is still highly beneficial to utilize symbolic features, especially when they are clean and have complementary information. Table 1 shows an example in the ACE 2005 dataset, where our baseline SpanBERT model incorrectly predicts the highlighted event mentions to be coreferential. The event triggers are semantically similar, making it challenging for our model to distinguish. However, notice that the event {head out}$_{ev1}$ is mentioned as if it was a real occurrence, and so its modality attribute is ASSERTED (LDC, 2005). In contrast, because of the phrase "were set to", we can infer that the event {leave}$_{ev2}$ did not actually happen (i.e., its modality attribute is OTHER). Therefore, our model should be able to avoid the mistake if it utilizes additional symbolic features such as the modality attribute in this case.

There are several previous methods that use contextual embeddings together with type-based or argument-based information (Lu et al., 2020; Yu et al., 2020). For example, Lu et al. (2020) proposes a new mechanism to better exploit event type information for coreference resolution. Despite their impressive performance, these methods are specific to one particular type of additional information.

In this paper, we propose general and effective methods for incorporating a wide range of sym-

[1]The code is publicly available at `https://github.com/laituan245/eventcoref`.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3491–3499
June 6–11, 2021.

bolic features into event coreference resolution. Simply concatenating symbolic features with contextual embeddings is not optimal, since the features can be noisy and contain errors. Also, depending on the context, some features can be more informative than others. Therefore, we design a novel context-dependent gated module to extract information from the symbolic features selectively. Combined with a simple regularization method that randomly adds noise into the features during training, our best models achieve state-of-the-art results on ACE 2005 (Walker et al., 2006) and KBP 2016 (Mitamura et al., 2016) datasets. To the best of our knowledge, our work is the first to explicitly focus on dealing with various noisy symbolic features for event coreference resolution.

2 Methods

2.1 Preliminaries

We focus on within-document event coreference resolution. The input to our model is a document D consisting of n tokens and k (predicted) event mentions $\{m_1, m_2, \ldots, m_k\}$. For each m_i, we denote the start and end indices of its trigger by s_i and e_i respectively. We assume the mentions are ordered based on s_i (i.e., If $i \leq j$ then $s_i \leq s_j$).

We also assume each m_i has K (predicted) categorical features $\{c_i^{(1)}, c_i^{(2)}, \ldots, c_i^{(K)}\}$, with each $c_i^{(u)} \in \{1, 2, \ldots, N_u\}$ taking one of N_u different discrete values. Table 2 lists the symbolic features we consider in this work. The definitions of the features and their possible values are in ACE and Rich ERE guidelines (LDC, 2005; Mitamura et al., 2016). The accuracy scores of the symbolic feature predictors are also shown in Table 2. We use OneIE (Lin et al., 2020) to identify event mentions along with their subtypes. For other symbolic features, we train a joint classification model based on SpanBERT. The appendix contains more details.

2.2 Single-Mention Encoder

Given a document D, our model first forms a contextualized representation for each input token using a Transformer encoder (Joshi et al., 2020). Let $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ be the output of the encoder, where $\mathbf{x}_i \in \mathbb{R}^d$. Then, for each mention m_i, its trigger's representation $\mathbf{t}_i$ is defined as the average of its token embeddings:

$$\mathbf{t}_i = \sum_{j=s_i}^{e_i} \frac{\mathbf{x}_j}{e_i - s_i + 1} \quad (1)$$

Dataset	Features	Acc. (Train)	Acc. (Dev)	Acc. (Test)
ACE 2005	Type	0.999	0.945	0.953
	Polarity	0.999	0.994	0.988
	Modality	0.999	0.856	0.884
	Genericity	0.999	0.865	0.872
	Tense	0.984	0.802	0.763
KBP 2016	Type	0.960	0.874	0.818
	Realis	0.979	0.845	0.840

Table 2: Symbolic features we consider in this work.

Next, by using K trainable embedding matrices, we convert the symbolic features of m_i into K vectors $\{\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)}, \ldots, \mathbf{h}_i^{(K)}\}$, where $\mathbf{h}_i^{(u)} \in \mathbb{R}^l$.

2.3 Mention-Pair Encoder and Scorer

Given two event mentions m_i and m_j, we define their trigger-based pair representation as:

$$\mathbf{t}_{ij} = \text{FFNN}_\text{t}\big(\big[\mathbf{t}_i, \mathbf{t}_j, \mathbf{t}_i \circ \mathbf{t}_j\big]\big) \quad (2)$$

where FFNN_t is a feedforward network mapping from $\mathbb{R}^{3\times d} \rightarrow \mathbb{R}^p$, and $\circ$ is element-wise multiplication. Similarly, we can compute their feature-based pair representations $\{\mathbf{h}_{ij}^{(1)}, \mathbf{h}_{ij}^{(2)}, \ldots, \mathbf{h}_{ij}^{(K)}\}$ as follows:

$$\mathbf{h}_{ij}^{(u)} = \text{FFNN}_\text{u}\big(\big[\mathbf{h}_i^{(u)}, \mathbf{h}_j^{(u)}, \mathbf{h}_i^{(u)} \circ \mathbf{h}_j^{(u)}\big]\big) \quad (3)$$

where $u \in \{1, 2, \ldots, K\}$, and FFNN_u is a feedforward network mapping from $\mathbb{R}^{3\times l} \rightarrow \mathbb{R}^p$.

Now, the most straightforward way to build the final pair representation $\mathbf{f}_{ij}$ of m_i and m_j is to simply concatenate the trigger-based representation and all the feature-based representations together:

$$\mathbf{f}_{ij} = [\mathbf{t}_{ij}, \mathbf{h}_{ij}^{(1)}, \mathbf{h}_{ij}^{(2)}, \ldots, \mathbf{h}_{ij}^{(K)}] \quad (4)$$

However, this approach is not always optimal. First, as the symbolic features are predicted, they can be noisy and contain errors. The performance of most symbolic feature predictors is far from perfect (Table 2). Also, depending on the specific context, some features can be more useful than others.

Inspired by studies on gated modules (Lin et al., 2019; Lai et al., 2019), we propose **Context-Dependent Gated Module** (CDGM), which uses a gating mechanism to extract information from the input symbolic features selectively (Figure 1). Given two mentions m_i and m_j, we use their trigger feature vector $\mathbf{t}_{ij}$ as the main controlling context to compute the filtered representation $\overline{\mathbf{h}}_{ij}^{(u)}$:

$$\overline{\mathbf{h}}_{ij}^{(u)} = \text{CDGM}^{(u)}\big(\mathbf{t}_{ij}, \mathbf{h}_{ij}^{(u)}\big) \quad (5)$$

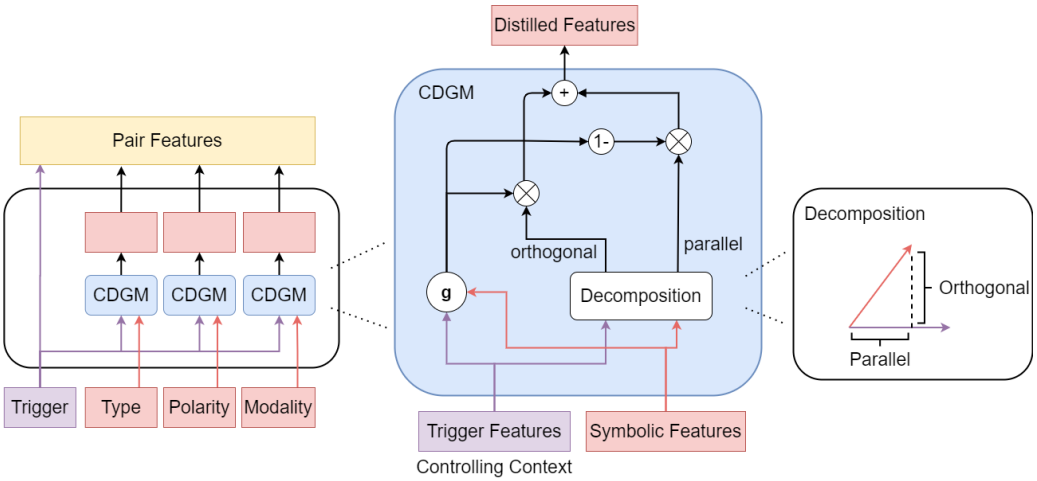

Figure 1: Overall architecture of our mention-pair encoder, which uses CDGMs to incorporate symbolic features.

where $u \in \{1, 2, \ldots, K\}$. More specifically:

$$\begin{aligned} \mathbf{g}_{ij}^{(u)} &= \sigma\big(\text{FFNN}_{\text{g}}^{(u)}\big([\mathbf{t}_{ij}, \mathbf{h}_{ij}^{(u)}]\big)\big) \\ \mathbf{o}_{ij}^{(u)}, \mathbf{p}_{ij}^{(u)} &= \text{DECOMPOSE}\big(\mathbf{t}_{ij}, \mathbf{h}_{ij}^{(u)}\big) \\ \overline{\mathbf{h}}_{ij}^{(u)} &= \mathbf{g}_{ij}^{(u)} \circ \mathbf{o}_{ij}^{(u)} + \big(1 - \mathbf{g}_{ij}^{(u)}\big) \circ \mathbf{p}_{ij}^{(u)} \end{aligned} \tag{6}$$

where σ denotes sigmoid function. $\text{FFNN}_{\text{g}}^{(u)}$ is a mapping from $\mathbb{R}^{2\times p} \rightarrow \mathbb{R}^{p}$. At a high level, $\mathbf{h}_{ij}^{(u)}$ is decomposed into an orthogonal component and a parallel component, and $\overline{\mathbf{h}}_{ij}^{(u)}$ is simply the fusion of these two components. In order to find the optimal mixture, $\mathbf{g}_{ij}$ is used to control the composition. The decomposition unit is defined as:

$$\begin{aligned} \textit{Parallel} \quad & \mathbf{p}_{ij}^{(u)} = \frac{\mathbf{h}_{ij}^{(u)} \cdot \mathbf{t}_{ij}}{\mathbf{t}_{ij} \cdot \mathbf{t}_{ij}} \mathbf{t}_{ij} \\ \textit{Orthogonal} \quad & \mathbf{o}_{ij}^{(u)} = \mathbf{h}_{ij}^{(u)} - \mathbf{p}_{ij}^{(u)} \end{aligned} \tag{7}$$

where $\cdot$ denotes dot product. The parallel component $\mathbf{p}_{ij}^{(u)}$ is the projection of $\mathbf{h}_{ij}^{(u)}$ on $\mathbf{t}_{ij}$. It can be viewed as containing information that is already part of $\mathbf{t}_{ij}$. In contrast, $\mathbf{o}_{ij}^{(u)}$ is orthogonal to $\mathbf{t}_{ij}$, and so it can be viewed as containing *new* information. Intuitively, when the original symbolic feature vector $\mathbf{h}_{ij}^{(u)}$ is very clean and has complementary information, we want to utilize the new information in $\mathbf{o}_{ij}^{(u)}$ (i.e., we want $\mathbf{g}_{ij}^{(u)} \approx \mathbf{1}$), and vice versa.

Finally, after using CDGMs to distill symbolic features, the final pair representation $\mathbf{f}_{ij}$ of m_i and m_j can be computed as follows:

$$\mathbf{f}_{ij} = [\mathbf{t}_{ij}, \overline{\mathbf{h}}_{ij}^{(1)}, \overline{\mathbf{h}}_{ij}^{(2)}, \ldots, \overline{\mathbf{h}}_{ij}^{(K)}] \tag{8}$$

And the coreference score $s(i, j)$ of m_i and m_j is:

$$s(i, j) = \text{FFNN}_{\text{a}}(\mathbf{f}_{ij}) \tag{9}$$

where FFNN_{a} is a mapping from $\mathbb{R}^{(K+1)\times p} \rightarrow \mathbb{R}$.

2.4 Training and Inference

Algorithm 1: Noise Addition for Symbolic Features

Input: Document D
Hyperparameters: $\{\epsilon_1, \epsilon_2, \cdots, \epsilon_K\}$
for $i = 1 \ \ldots \ k$ **do**
  for $u = 1 \ \ldots \ K$ **do**
    With prob. ϵ_u, replace $c_i^{(u)}$ by $\hat{c}_i^{(u)} \sim \textit{Uniform}(N_u)$
  end
end

Training We use the same loss function as in (Lee et al., 2017). Also, notice that the training accuracy of a feature predictor is typically much higher than its accuracy on the dev/test set (Table 2). If we simply train our model without any regularization, our CDGMs will rarely come across noisy symbolic features during training. Therefore, to encourage our CDGMs to actually learn to distill reliable signals, we also propose a simple but effective **noisy training** method. Before passing a training data batch to the model, we randomly add noise to the predicted features. More specifically, for each document D in the batch, we go through every symbolic feature of every event mention in D and consider sampling a new value for the feature. The operation is described in Algorithm 1 (we use the same notations mentioned in Section 2.1). $\{\epsilon_1, \epsilon_2, \cdots, \epsilon_K\}$ are hyperparameters determined by validation. In general, the larger the discrepancy between the train and test accuracies, the larger ϵ.

Inference For each (predicted) mention m_i, our model will assign an antecedent a_i from all pre-

ACE (Cross-Validation)	CoNLL	AVG
SSED + Supervised$_{\text{Extended}}$ (2016)	55.23	52.53
SSED + MSEP (2016)	53.80	51.38
ACE (Test Data)	**CoNLL**	**AVG**
Baseline	58.93	55.78
Simple (All Features)	57.55	54.79
CDGM (All Features)	58.99	56.32
Noise (All Features)	60.43	57.85
CDGM + Noise (All Features)	**62.07**	**59.76**

Table 3: End-to-end results on ACE 2005 (using predicted triggers and predicted symbolic features).

System	CoNLL	AVG
UTD's system (2015)	32.69	30.08
Joint Learning (2017b)	35.77	33.08
E^{3}C (2020)	41.97	38.66
Baseline	40.57	37.59
Simple (All Features)	41.40	38.58
CDGM + Noise (All Features)	**43.55**	**40.61**

Table 4: End-to-end results on KBP 2016 (using predicted triggers and predicted symbolic features).

ceding mentions or a dummy antecedent ϵ: $a_i \in Y(i) = \{\epsilon, m_1, m_2 \ldots, m_{i-1}\}$. Basically, $a_i = \arg\max_{j<i} s(i,j)$. The dummy antecedent ϵ represents two possible cases: (1) m_i is not actually an event mention (2) m_i is indeed an event mention but it is not coreferent with any previous extracted mentions. In addition, we fix $s(i,\epsilon)$ to be 0.

3 Experiments and Results

Data and Experiments Setup We evaluate our methods on two English datasets: ACE2005 (Walker et al., 2006) and KBP2016 (Ji et al., 2016; Mitamura et al., 2016). We report results in terms of F1 scores obtained using the CoNLL and AVG metrics. By definition, these metrics are the summary of other standard coreference metrics, including B^3, MUC, $CEAF_e$, and BLANC (Lu and Ng, 2018). We use SpanBERT (spanbert-base-cased) as the Transformer encoder (Wolf et al., 2020a; Joshi et al., 2020). More details about the datasets and hyperparameters are in the appendix. We refer to models that use only trigger features as [Baseline]. In a baseline model, $\mathbf{f}_{ij}$ is simply $\mathbf{t}_{ij}$ (Eq. 2). We refer to models that use only the simple concatenation strategy as [Simple] (Eq. 4), and models that use the simple concatenation strategy and the noisy training method as [Noise].

Overall Results (on Predicted Mentions) Table 3 and Table 4 show the overall end-to-end results on ACE2005 and KBP2016, respectively. We use OneIE (Lin et al., 2020) to extract event mentions and their types. Other features are predicted by a simple Transformer model. Overall, our full model outperforms the baseline model by a large margin and significantly outperforms state-of-the-art on KBP 2016. Our ACE 2005 scores are not directly comparable with previous work, as Peng et al. (2016) conducted 10-fold cross-validation and essentially used more training data. Nevertheless, the magnitude of the differences in scores between our best model and the state-of-the-art methods indicates the effectiveness of our methods.

ACE (Test Data)	CoNLL	AVG
PAIREDRL (2020)	84.65	-
Baseline	81.62	81.49
Simple (All Features)	75.32	74.94
CDGM + Noise (All Features)	**84.76**	**83.95**

Table 5: Results on ACE 2005 using gold triggers and predicted symbolic features.

ACE (Test Data)	CoNLL	AVG
Baseline	81.62	81.49
Simple (All Features)	85.75	85.40
CDGM (All Features)	**87.90**	**88.30**
CDGM + Noise (All Features)	85.40	85.38

Table 6: Results on ACE 2005 using gold triggers and ground-truth symbolic features.

Overall Results (on Ground-truth Triggers) The overall results on ACE 2005 using ground-truth triggers and predicted symbolic features are shown in Table 5. The performance of our full model is comparable with previous state-of-the-art result in (Yu et al., 2020). To better analyze the usefulness of symbolic features as well as the effectiveness of our methods, we also conduct experiments using *ground-truth* triggers and *ground-truth* symbolic features (Table 6). First, when the symbolic features are clean, incorporating them using the simple concatenation strategy can already boost the performance significantly. The symbolic features contain information complementary to that in the SpanBERT contextual embeddings. Second, we also see that the noisy training method is not helpful when the symbolic features are clean. Unlike other regularization methods such as dropout (Srivastava et al., 2014) and weight decay (Krogh and Hertz, 1992), the main role of our noisy training method is not to reduce overfitting in the traditional sense. Its main function is to help CDGMs learn to distill reliable signals from noisy features.

Features	AVG (Simple)	AVG (CDGM + Noise)	Δ_{AVG}
Subtype	56.41	57.02	**+0.61**
Polarity	56.06	57.03	**+0.97**
Modality	54.81	58.54	**+3.73**
Genericity	54.70	57.82	**+3.12**
Tense	54.28	56.62	**+2.34**

Table 7: Impact of Symbolic Features (ACE 2005)

Impact of Different Symbolic Features Table 7 shows the results of incorporating different types of symbolic features on the ACE 2005 dataset. Overall, our methods consistently perform better than the simple concatenation strategy across all feature types. The gains are also larger for more noisy features than clean features (feature prediction accuracies were shown in Table 2). This suggests that our methods are particularly useful in situations where the symbolic features are noisy.

Comparison with Multi-Task Learning We also investigate whether we can incorporate symbolic semantics into coreference resolution by simply doing multi-task training. We train our baseline model to jointly perform coreference resolution and symbolic feature prediction. The test AVG score on ACE 2005 is only 56.5. In contrast, our best model achieves an AVG score of 59.76 (Table 3).

Qualitative Examples Table 8 shows few examples from the ACE 2005 dataset that illustrate how incorporating symbolic features using our proposed methods can improve the performance of event coreference resolution. In each example, our baseline model incorrectly predicts the highlighted event mentions to be coreferential.

Remaining Challenges Previous studies suggest that there exist different types and degrees of event coreference (Recasens et al., 2011; Hovy et al., 2013). Many methods (including ours) focus on the full strict coreference task, but other types of coreference such as partial coreference have remained underexplored. Hovy et al. (2013) defines two core types of partial event coreference relations: subevent relations and membership relations. Subevent relations form a stereotypical sequence of events, whereas membership relations represent instances of an event collection. We leave tackling the partial coreference task to future work.

4 Related Work

Several previous approaches to within-document event coreference resolution operate by first applying a mention-pair model to compute pairwise distances between event mentions, and then they apply a clustering algorithm such as agglomerative clustering or spectral graph clustering (Chen et al., 2009; Chen and Ji, 2009; Chen and Ng, 2014; Nguyen et al., 2016; Huang et al., 2019). In addition to trigger features, these methods use a variety of additional symbolic features such as event types, attributes, arguments, and distance. These approaches do not use contextual embeddings such as BERT and SpanBERT (Devlin et al., 2019; Joshi et al., 2020). Recently, there are several studies that use contextual embeddings together with type-based or argument-based information (Lu et al., 2020; Yu et al., 2020). These methods design networks or mechanisms that are specific to only one type of symbolic features. In contrast, our work is more general and can be effectively applied to a wide range of symbolic features.

... {Negotiations}$_{ev1}$ between Washington and ...
... think that this will affect the {elections}$_{ev2}$ unless ...
ev1 (`Contact:Meet`)
ev2 (`Personnel:Elect`)
... since you are **not** directly {elected}$_{ev1}$, it would be ...
... Az-Zaman daily that {elections}$_{ev2}$ should be held ...
ev1 (`Personnel:Elect`): Polarity = NEGATIVE
ev2 (`Personnel:Elect`): Polarity = POSITIVE
... told reporters after his {appeal}$_{ev1}$ was rejected ...
... most junior of the court of {appeal}$_{ev2}$, and its ...
ev1 (`Justice:Appeal`): Genericity = SPECIFIC
ev2 (`Justice:Appeal`): Genericity = GENERIC

Table 8: Examples of using symbolic features to improve event coreference resolution.

5 Conclusions and Future Work

In this work, we propose a novel gated module to incorporate symbolic semantics into event coreference resolution. Combined with a simple noisy training technique, our best models achieve competitive results on ACE 2005 and KBP 2016. In the future, we aim to extend our work to address more general problems such as cross-lingual cross-document coreference resolution.

Acknowledgement

This research is based upon work supported in part by U.S. DARPA KAIROS Program No. FA8750-19-2-1004, U.S. DARPA AIDA Program No. FA8750-18-2-0014, and Air Force No. FA8650-17-C-7715. The views and conclusions contained herein are those of the authors and should not be

interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

C. Chen and Vincent Ng. 2014. Sinocoreferencer: An end-to-end chinese event coreference resolver. In *LREC*.

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 Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2913–2920. AAAI Press.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. 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 167–176, Beijing, China. Association for Computational Linguistics.

Zheng Chen and Heng Ji. 2009. Graph-based event coreference resolution. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing (TextGraphs-4)*, pages 54–57, Suntec, Singapore. Association for Computational Linguistics.

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, Borovets, Bulgaria. Association for Computational Linguistics.

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, Copenhagen, Denmark. Association for Computational Linguistics.

Prafulla Kumar Choubey, Kaushik Raju, and Ruihong Huang. 2018. Identifying the most dominant event in a news article by mining event coreference relations. 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 340–345, New Orleans, Louisiana. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chase Duncan, Liang-Wei Chan, Haoruo Peng, Hao Wu, Shyam Upadhyay, Nitish Gupta, Chen-Tse Tsai, Mark Sammons, and Dan Roth. 2017. UI CCG TAC-KBP2017 submissions: Entity discovery and linking, and event nugget detection and co-reference. In *Proceedings of the 2017 Text Analysis Conference, TAC 2017, Gaithersburg, Maryland, USA, November 13-14, 2017*. NIST.

C. Harris, K. J. Millman, S. Walt, Ralf Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, Nathaniel J. Smith, R. Kern, Matti Picus, S. Hoyer, M. Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'io, Mark Wiebe, P. Peterson, Pierre G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, Christoph Gohlke, and T. E. Oliphant. 2020. Array programming with numpy. *Nature*, 585 7825:357–362.

Eduard Hovy, Teruko Mitamura, Felisa Verdejo, Jun Araki, and Andrew Philpot. 2013. Events are not simple: Identity, non-identity, and quasi-identity. In *Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 21–28, Atlanta, Georgia. Association for Computational Linguistics.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1148–1158, Portland, Oregon, USA. Association for Computational Linguistics.

Heng Ji, Joel Nothman, H Trang Dang, and Sydney Informatics Hub. 2016. Overview of tac-kbp2016 tri-lingual edl and its impact on end-to-end cold-start kbp. *Proceedings of TAC*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8(0):64–77.

Anders Krogh and John Hertz. 1992. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann.

Tuan Lai, Quan Hung Tran, Trung Bui, and Daisuke Kihara. 2019. A gated self-attention memory network for answer selection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5953–5959, Hong Kong, China. Association for Computational Linguistics.

LDC. 2005. English annotation guidelines for events version 5.4. 3. *Linguistic Data Consortium*, 1.

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, Copenhagen, Denmark. Association for Computational Linguistics.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Ying Lin, Liyuan Liu, Heng Ji, Dong Yu, and Jiawei Han. 2019. Reliability-aware dynamic feature composition for name tagging. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 165–174, Florence, Italy. Association for Computational Linguistics.

J. Lu and Vincent Ng. 2017a. Learning antecedent structures for event coreference resolution. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 113–118.

Jing Lu and Vincent Ng. 2015. Utd's event nugget detection and coreference system at KBP 2015. In *Proceedings of the 2015 Text Analysis Conference, TAC 2015, Gaithersburg, Maryland, USA, November 16-17, 2015, 2015*. NIST.

Jing Lu and Vincent Ng. 2016. Event coreference resolution with multi-pass sieves. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3996–4003, Portorož, Slovenia. European Language Resources Association (ELRA).

Jing Lu and Vincent Ng. 2017b. 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, Vancouver, Canada. Association for Computational Linguistics.

Jing Lu and Vincent Ng. 2018. Event coreference resolution: A survey of two decades of research. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5479–5486. International Joint Conferences on Artificial Intelligence Organization.

Yaojie Lu, Hongyu Lin, Jialong Tang, Xianpei Han, and Le Sun. 2020. End-to-end neural event coreference resolution. *arXiv preprint arXiv:2009.08153*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Teruko Mitamura, Zhengzhong Liu, and Eduard H. Hovy. 2016. Overview of TAC-KBP 2016 event nugget track. In *Proceedings of the 2016 Text Analysis Conference, TAC 2016, Gaithersburg, Maryland, USA, November 14-15, 2016*. NIST.

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, TAC 2016, Gaithersburg, Maryland, USA, November 14-15, 2016*. NIST.

Adam Paszke, S. Gross, Francisco Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, E. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, B. Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *ArXiv*, abs/1912.01703.

Haoruo Peng, Yangqiu 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, Austin, Texas. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

M. Recasens, E. Hovy, and M. Martí. 2011. Identity, non-identity, and near-identity: Addressing the complexity of coreference. *Lingua*, 121:1138–1152.

Mark Sammons, Haoruo Peng, Yangqiu Song, Shyam Upadhyay, Chen-Tse Tsai, Pavankumar Reddy, Subhro Roy, and Dan Roth. 2015. Illinois CCG TAC 2015 event nugget, entity discovery and linking, and slot filler validation systems. In *Proceedings of the*

2015 Text Analysis Conference, TAC 2015, Gaithersburg, Maryland, USA, November 16-17, 2015, 2015. NIST.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Lucy Vanderwende, Michele Banko, and Arul Menezes. 2004. Event-centric summary generation. In *Working notes of the Document Understanding Conference 2004*. ACL.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57:45.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020a. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020b. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xiaodong Yu, Wenpeng Yin, and Dan Roth. 2020. Paired representation learning for event and entity coreference. *arXiv preprint arXiv:2010.12808*.

Tongtao Zhang, Hongzhi Li, Heng Ji, and Shih-Fu Chang. 2015. Cross-document event coreference resolution based on cross-media features. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP2015)*.

A Appendix

Section A.1 describes our symbolic feature predictors. Section A.2 provides the details of the datasets we used. Section A.3 describes the hyperparameters and their value ranges that were explored. Section A.4 presents our reproducibility checklist.

Dataset	Train (# Docs)	Dev (# Docs)	Test (# Docs)
ACE-2005	529	30	40
KBP 2016	509	139	169

Table 9: Basic statistics of the datasets.

A.1 Symbolic Feature Predictors

In an end-to-end setting, we train and use OneIE (Lin et al., 2020) to identify event mentions along with their subtypes. For other symbolic features, we train a simple joint model. More specifically, given a document, our joint model first forms contextualized representations for the input tokens using SpanBERT (Joshi et al., 2020). Each event mention's representation is then defined as the average of the embeddings of the tokens in its trigger. After that, we feed the mentions' representations into classification heads for feature value prediction. Each classification head is a standard multi-layer feedforward network with softmax output units.

The event detection performance of OneIE on the test set of ACE 2005 is 74.7 (Type-F1 score). OneIE's performance on KBP 2016 is 55.20 (Type-F1 score). For reference, the performance of the event detection component of E^3C (Lu et al., 2020) on KBP 2016 is 55.38 (Type-F1 score).

A.2 Datasets Description

In this work, we use two English within-document coreference datasets: ACE 2005 and KBP 2016. The ACE 2005 English corpus contains fine-grained event annotations for 599 articles from a variety of sources. We use the same split as that stated in (Chen et al., 2015), where there are 529/30/40 documents in the train/dev/test split. In ACE, a strict notion of event coreference is adopted, which requires two event mentions to be coreferential if and only if they had the same agent(s), patient(s), time, and location. For KBP 2016, we follow the setup of (Lu and Ng, 2017a), where there are 648 documents that can be used for training and 169 documents for testing. We train our model on 509 documents randomly chosen from the training documents and tune parameters on the remaining 139 training documents. Different from ACE, KBP adopts a more relaxed definition of event coreference, where two event mentions can be coreferent as long as they intuitively refer to the same real-world event. Table 9 summarizes the basic statistics of the datasets.

A.3 Hyperparameters

We use SpanBERT (spanbert-base-cased) as the Transformer encoder (Wolf et al., 2020a; Joshi et al., 2020). We did hyperparameter tuning using the datasets' dev sets. For all the experiments, we pick the model which achieves the best AVG score on the dev set, and then evaluate it on the test set. For each of our models, two different learning rates are used, one for the lower pretrained Transformer encoder and one for the upper layer. The optimal hyperparameter values are variant-specific, and we experimented with the following range of possible values: $\{8, 16\}$ for batch size, {3e-5, 4e-5, 5e-5} for lower learning rate, {1e-4, 2.5e-4, 5e-4} for upper learning rate, and $\{50, 100\}$ for number of training epochs. Table 10 shows the value of ϵ we used for each symbolic feature type. In general, the larger the discrepancy between the train and test accuracies, the larger the value of ϵ.

Dataset	Features	ϵ (predicted mentions)	ϵ (gold mentions)
ACE 2005	Type	0.00	0.10
	Polarity	0.00	0.02
	Modality	0.15	0.20
	Genericity	0.15	0.20
	Tense	0.25	0.30
KBP 2016	Type	0.05	-
	Realis	0.10	-

Table 10: The value of ϵ for each feature type.

A.4 Reproducibility Checklist

We present the reproducibility information of the paper. Due to license reason, we cannot provide downloadable links for ACE 2005 and KBP 2016.

Implementation Dependencies Libraries Pytorch 1.6.0 (Paszke et al., 2019), Transformers 3.0.2 (Wolf et al., 2020b), Numpy 1.19.1 (Harris et al., 2020), CUDA 10.2.

Computing Infrastructure The experiments were conducted on a server with Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz and NVIDIA Tesla V100 GPUs. The allocated RAM is 187G. GPU memory is 16G.

Average Runtime Table 11 shows the estimated average run time of our full model.

Number of Model Parameters The number of parameters in a baseline model is about 109.7M parameters. The number of parameters in a full model trained on the KBP 2016 dataset is about 111.4M parameters. The number of parameters in a full model trained on the ACE 2005 dataset is about 113.8M parameters.

Dataset	One Training Epoch	Evaluation (Dev Set)	Evaluation (Test Set)
ACE 2005	65.5 seconds	2.3 seconds	2.4 seconds
KBP 2016	103.6 seconds	10.7 seconds	11.8 seconds

Table 11: Estimated average runtime of our full model.

Hyperparameters of Best-Performing Models Table 12 summarizes the hyperparameter configurations of best-performing models. Note that Table 10 already showed the hyperparameters used for the noisy training method.

Hyperparameters	ACE 2005 (end-to-end)	KBP 2016 (end-to-end)	ACE 2005 (gold mentions)
Symbolic Features Used	All (CDGM)	All (CDGM)	All (CDGM)
Noisy Training	Yes	Yes	Yes
Lower Learning Rate	4e-5	5e-5	5e-5
Upper Learning Rate	2.5e-4	5e-4	5e-4
Batch Size	16	8	8
Number Epochs	100	50	50

Table 12: Hyperparameters for best-performing models (refer to Table 10 for the hyperparameters used for the noisy training method).

Expected Validation Performance We repeat training five times for each best-performing model. We show the average validation performance in Table 13. Our validation scores on KBP 2016 are not comparable to that of (Lu et al., 2020), because we split the original 648 training documents into the final train set and dev set randomly. We still use the same test set. For each best-performing model, we report the test performance of the checkpoint with the best AVG score in the main paper.

Dataset	Avg. CoNLL score	Avg. AVG score
ACE 2005 (end-to-end)	60.20	58.80
KBP 2016 (end-to-end)	54.86	48.83
ACE 2005 (gold mentions)	81.9	83.02

Table 13: Average validation performance.