A Data

A.1 Dataset statistics

Table 8 provides summary statistics for all datasets used in the paper.

	Domain	Docs	Ent	Rel	Trig	Arg
ACE05	News	511	7	6	-	-
ACE05-E	News	599	7	-	33	22
SciERC	AI	500	6	7	-	-
GENIA	Biomed	1999	5	-	-	-
WLP	Bio lab	622	18	13	-	-

Table 8: Datasets for joint entity and relation extraction and their statistics. *Ent*: Number of entity categories. *Rel*: Number of relation categories. *Trig*: Number of event trigger categories. *Arg*: Number of event argument categories.

A.2 ACE event data preprocessing and evaluation

There is some inconsistency in the ACE event literature on how to handle "time" and "value" event arguments, which are not technically named entities. Some authors, for instance Yang and Mitchell (2016) leave them in and create new entity types for them. We follow the preprocessing of Zhang et al. (2019), who ignore these arguments entirely, since these authors shared their preprocessing code with us and report the current state of the art. We will be releasing code at `https://github.com/dwadden/dygiepp` to exactly reproduce our data preprocessing, so that other authors can compare their approaches on our data. Due to this discrepancy in the literature, however, our results for named entity and event argument classification are not directly comparable with some previous works.

In addition, there is some confusion on what constitutes an "Event argument identification". Following Yang and Mitchell (2016) and Zhang et al. (2019), we say that an argument is identified correctly if its offsets and event type are correct. Some other works seem to require require only that an argument's offsets be identified, not its event type. We do not compare against these.

B Graph Propagation

We model relation and coreference interactions similarly to Luan et al. (2019), and extend the approach to incorporate events. We detail the event propagation procedure here. While the relation and coreference span graphs consist of a single type of node, the event graph consists of two types of nodes: triggers and arguments.

The intuition behind the event graph is to provide each trigger with information about its potential arguments, and each potential argument with information about triggers for the events in which it might participate.[2]

The model iterates between updating the triggers based on the representations of their likely arguments, and updating the arguments based on the representations of their likely triggers. More formally, denote the number of possible semantic roles played by an event argument (i.e. the number of argument labels) as L_A, B_T as a beam of candidate trigger tokens, and B_A as a beam of candidate argument spans. These beams are selected by learned scoring functions. For each trigger $\mathbf{h}_i^t \in B_T$ and argument $\mathbf{g}_j^t \in B_A$, the model computes a similarity vector $\mathbf{V}_A^t(i,j)$ by concatenating the trigger and argument embeddings and running them through a feedforward neural network. The k^{th} element of $\mathbf{V}_A^t(i,j)$ scores how likely it is that argument $\mathbf{g}_j$ plays role k in the event triggered by $\mathbf{h}_i$.

Extending Equation 1, the model updates the trigger $\mathbf{h}_i$ by taking an average of the candidate argument embeddings, weighted by the likelihood that each candidate plays a role in the event:

$$\mathbf{u}_{A\to T}^t(i) = \sum_{j\in B_A} \mathbf{A}_{A\to T} f(\mathbf{V}_A^t(i,j)) \odot \mathbf{g}_j^t, \quad (2)$$

where $\mathbf{A}_{A\to T} \in \mathbb{R}^{d\times L_A}$ is a learned projection matrix, f is a ReLU function, $\odot$ is an element-wise product, and d is the dimension of the span embeddings. Then, the model computes a gate determining how much of the update from (2) to apply to the trigger embedding:

$$f_{A\to T}^t(i) = \sigma\left(\mathbf{W}_{A\to T}[\mathbf{h}_i^t, \mathbf{u}_{A\to T}^t(i)]\right), \quad (3)$$

where $\mathbf{W}_{A\to T} \in \mathbb{R}^{d\times 2d}$ is a learned projection matrix and σ is the logistic sigmoid function. Finally, the updated trigger embeddings are computed as follows:

$$\mathbf{h}_i^{t+1} = f_{A\to T}^t(i)\odot\mathbf{h}_i^t + (1 - f_{A\to T}^t(i))\odot\mathbf{u}_{A\to T}^t(i). \quad (4)$$

Similarly, an update for argument span $\mathbf{g}_j$ is computed via messages $\mathbf{u}_{T\to A}^t(j)$ as a weighted

[2]Event propagation is a somewhat different idea from (Sha et al., 2018), who model argument-argument interactions using a learned tensor. We experimented with adding a similar tensor to our architecture, but did not see any clear performance improvements.

average over the trigger spans. The update is computed analgously to Equation 2, with a new trainable matrix $\mathbf{A}_{T\to A}$. Finally, the gate $f^t_{T\to A}(j)$ and the updated argument spans $\mathbf{g}_j^{t+1}$ are computed in the same fashion as (3) and (4) respectively. This process represents one iteration of event graph propagation. The outputs of the graph propagation are contextualized trigger and argument representations. When event propagation is performed, the final trigger scorer takes the contextualized surrogate spans $\mathbf{h}_i$ as input rather than the original token embeddings $\mathbf{d}_i$.

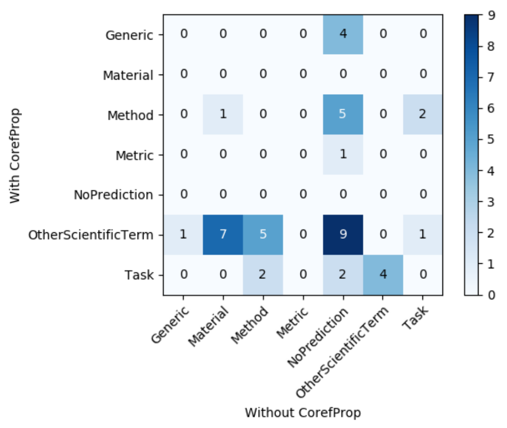
C `CorefProp` visualizations

Figure 3 shows confusion matrices for cases where `CorefProp` corrects and introduces a mistake on SciERC named entity recognition. It tends to correct mistakes where the base model either missed an entity, or was overly specific – classifying an entity as a *Material* or *Method* when it should have been classified with the more general label *OtherScientificTerm*. Similarly, `CorefProp` introduces mistakes by assigning labels that are too general, or by missing predictions.

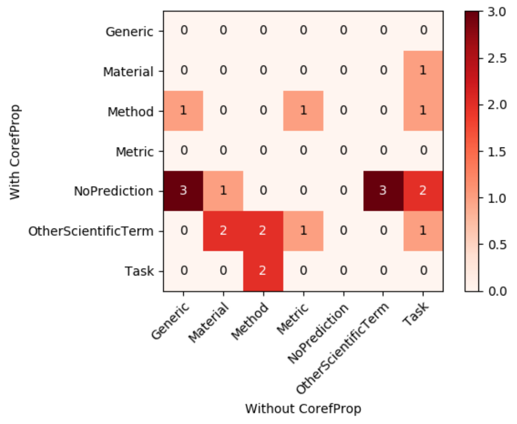
Figure 4 shows a visualization of the coreference attention weights for a named entity in the GENIA data set. The acronym “v-erbA” is correctly identified as a protein, due to a span update from its coreference antecedent “v-erbA oncoprotein”.

D Statistical significance of results

For a subset of the results in Table 1, we evaluated statistical significance by re-training a model with 5 random seeds and computing the mean and standard error of the mean of the F1 scores. For ensemble models (trigger detection), we trained 3 ensembles instead of 5 due to the large computational demands of training ensemble models. Due to the large number of experiments performed, it was impractical to perform these tests for every experiment. We report means and standard errors in Table 9. For most results, our mean is more than two standard errors above the previous state of the art; those results are significant. Our event argument results are not significant. For trigger classification, our mean F1 is a little less than two standard errors above the state of the art, indicating moderate significance.



(a) `CorefProp` corrects a mistake.



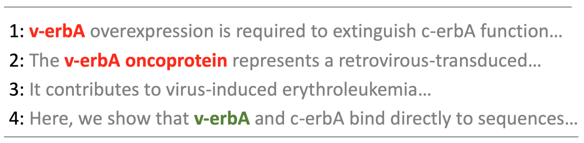
(b) `CorefProp` makes a mistake.

Figure 3: Confusion matrix of cases in the SciERC dev set where coreference propagation changed a prediction from incorrect to correct (Fig. 3a), or correct to incorrect (Fig. 3b). `CorefProp` leads to more corrections than mistakes, and tends to make less specific predictions (i.e. *OtherScientificTerm*).

E Implementation Details

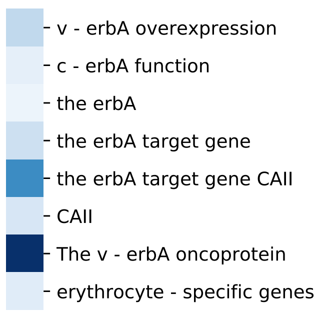
Learning rate schedules For BERT finetuning, we used BertAdam with a maximum learning rate of 1×10^{-3} for the task specific layers and 5×10^{-5} for BERT. For the learning rate schedule, we had an initial warmup period of 40000 batches for the BERT parameters, and 20000 batches for the task specific layers. Following the warmup period, we linearly decayed the learning rate.

For event extraction models with no finetuning we found that SGD with momentum performed better than Adam. We used the PyTorch implementation of SGD, with an initial learning rate of 0.02, momentum of 0.9, weight decay of 1×10^{-6} and a batch size of 15 sentences. We cut the learning rate in half whenever dev set F1 had not decreased



1: v-erbA overexpression is required to extinguish c-erbA function…
2: The v-erbA oncoprotein represents a retrovirous-transduced…
3: It contributes to virus-induced erythroleukemia…
4: Here, we show that v-erbA and c-erbA bind directly to sequences…

(a) To classify the mention of *v-erbA* in Sentence 4, the model can share information from its coreference antecedents (in red).



(b) Coreference attention weights. Darker is larger. The biggest update comes from *The v-erbA oncoprotein*

Figure 4: **`CorefProp` aggregates information from informative text spans**. By using the representation of the span *v-erbA oncoprotein* in Sentence 2 to update the representation of *v-erbA* in sentence 4, the model is able to correctly classify the latter entity mention as a *protein*.

for 3 epochs.

For all models, we used early stopping based on dev set loss.

Hyperparameter selection We experimented with both $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ on all tasks. We found that $\text{BERT}_{\text{LARGE}}$ provided improvement on event extraction with a final LSTM layer, but not on any of the other tasks or event extraction with BERT fine-tuning. In our final experiments we used $\text{BERT}_{\text{BASE}}$ except in the one setting mentioned were $\text{BERT}_{\text{LARGE}}$ was better.

We experiment with hidden layer sizes of 150, 300, and 600 for our feedforward scoring functions. We found that 150 worked well for all tasks except event extraction, where 600 hidden units were used.

Event extraction modeling details For event extraction we experimented with a final "decoding" module to ensure that event argument assignments were consistent with the types of the events in which they participated – for instance, an entity participating in a "Personnel.Nominate" event can play the role of "Agent", but not the role of "Prosecutor". However, we found in practice that the model learned which roles were compatible with each event type, and constrained decoding did not improve performance. For argument classification, we included the entity label of each candidate argument as an additional feature. At train time we used gold entity labels and at inference time we used the softmax scores for each entity class as predicted by the named entity recognition model.

Dataset	Task	SOTA	Ours (mean)	Ours (sem)
ACE05-Event*	Entity	87.1	90.4	0.1
	Trig-ID	73.9	76.1	0.4
	Trig-C	72.0	73.0	0.6
	Arg-ID	57.2	54.0	0.4
	Arg-C	52.4	51.3	0.4
SciERC	Entity	65.2	66.3	0.4
	Relation	41.6	46.2	0.4

Table 9: Mean and standard error of the mean. Trig-C is moderately significant. Arg-ID and Arg-C do not improve SOTA when averaging across five models. The remaining results are highly significant. Note that our means here differ from the numbers in Table 1, where we report our best single run to be consistent with previous literature.

Event model ensembling For the event extraction experiments summarized in Table 4 we performed early stopping based on dev set argument role classification performance. However, our trigger detector tended to overfit before the argument classifier had finished training. We also found stopping based on dev set error to be unreliable, due to the small size and domain shift between dev and test split on the ACE05-E data set. Therefore, for our final predictions reported in Table 1, we trained a four-model ensemble optimized for trigger detections rather than event argument classification, and combined the trigger predictions from this model with the argument role predictions from our original model. This combination improves both trigger detection *and* argument classification, since an argument classification is only correct if the trigger to which it refers is also classified correctly.