A Frustratingly Easy Approach for Entity and Relation Extraction

Zexuan Zhong **Danqi Chen**
Department of Computer Science
Princeton University
{zzhong, danqic}@cs.princeton.edu

Abstract

End-to-end relation extraction aims to identify named entities and extract relations between them. Most recent work models these two subtasks jointly, either by casting them in one structured prediction framework, or performing multi-task learning through shared representations. In this work, we present a simple pipelined approach for entity and relation extraction, and establish the new state-of-the-art on standard benchmarks (ACE04, ACE05 and SciERC), obtaining a 1.7%-2.8% absolute improvement in relation F1 over previous joint models with the same pre-trained encoders. Our approach essentially builds on two independent encoders and merely uses the entity model to construct the input for the relation model. Through a series of careful examinations, we validate the importance of learning distinct contextual representations for entities and relations, fusing entity information early in the relation model, and incorporating global context. Finally, we also present an efficient approximation to our approach which requires only one pass of both entity and relation encoders at inference time, achieving an 8-16× speedup with a slight reduction in accuracy.[1]

1 Introduction

Extracting entities and their relations from unstructured text is a fundamental problem in information extraction. This problem can be decomposed into two subtasks: named entity recognition (Sang and De Meulder, 2003; Ratinov and Roth, 2009) and relation extraction (Zelenko et al., 2002; Bunescu and Mooney, 2005). Early work employed a pipelined approach, training one model to extract entities (Florian et al., 2004, 2006), and another model to classify relations between them (Zhou et al., 2005; Kambhatla, 2004; Chan and Roth, 2011). More recently, however, end-to-end evaluations have been dominated by systems that model these two tasks jointly (Li and Ji, 2014; Miwa and Bansal, 2016; Katiyar and Cardie, 2017; Zhang et al., 2017a; Li et al., 2019; Luan et al., 2018, 2019; Wadden et al., 2019; Lin et al., 2020; Wang and Lu, 2020). There has been a long held belief that joint models can better capture the interactions between entities and relations and help mitigate error propagation issues.

In this work, we re-examine this problem and present a simple approach which learns *two* encoders built on top of deep pre-trained language models (Devlin et al., 2019; Beltagy et al., 2019; Lan et al., 2020). The two models — which we refer them as to the *entity model* and *relation model* throughout the paper — are trained independently and the relation model only relies on the entity model to provide input features. Our entity model builds on span-level representations and our relation model builds on contextual representations specific to *a given pair of spans*. Despite its simplicity, we find this pipelined approach to be extremely effective: using the same pre-trained encoders, our model outperforms all previous joint models on three standard benchmarks: ACE04, ACE05 and SciERC, advancing the previous state-of-the-art by 1.7%–2.8% absolute in relation F1.

To better understand the effectiveness of this approach, we carry out a series of careful analyses. We observe that, (1) the contextual representations for the entity and relation models essentially capture distinct information, so sharing their representations hurts performance; (2) it is crucial to fuse the entity information (both boundary and type) at the *input layer* of the relation model; (3) leveraging cross-sentence information is useful in both tasks. Hence, we expect that this simple model will serve as a very strong baseline in end-to-end relation extraction and make us rethink the value of joint modeling of entities and relations.

Finally, one possible shortcoming of our approach is that we need to run our relation model

[1] Our code and models are publicly available at https://github.com/princeton-nlp/PURE.

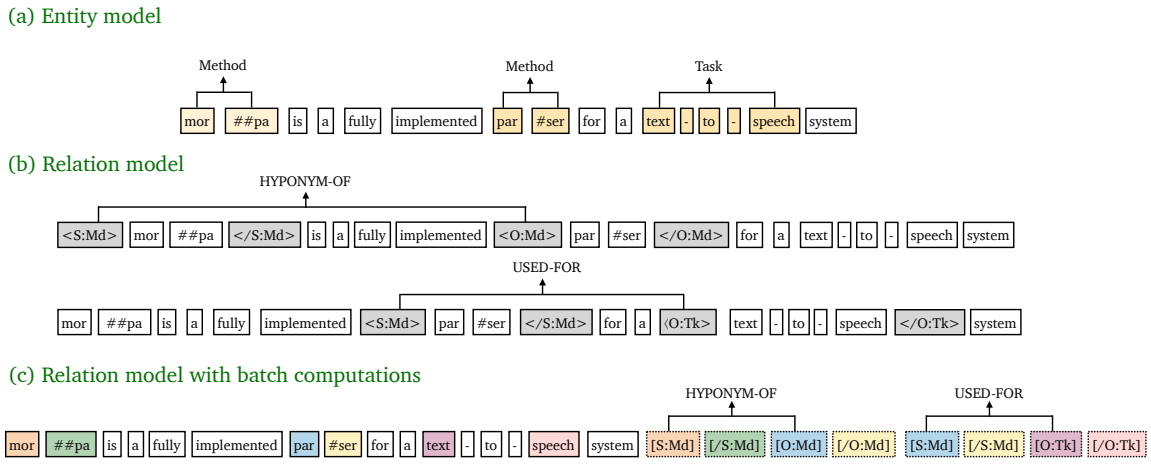

Figure 1: An example from the SciERC dataset (Luan et al., 2018). Given an input sentence *MORPA is a fully implemented parser for a text-to-speech system*, an end-to-end relation extraction system is expected to extract that MORPA and PARSER are entities of type METHOD, TEXT-TO-SPEECH is a TASK, as well as MORPA is a *hyponym of* PARSER and MORPA is *used for* TEXT-TO-SPEECH. (a) Our entity model predicts all the entities at once. (b) Our relation model considers every pair of entities independently by inserting typed entity markers (e.g., [S:MD]: the subject is a METHOD, [O:TK]: the object is a TASK). (c) We also proposed an approximation relation model which supports batch computations. The tokens of the same color share the positional embeddings (see Section 4.3 for more details).

once for every pair of entities. To alleviate this issue, we present a novel and efficient alternative by approximating and batching the computations for different groups of entity pairs at inference time. This approximation achieves an 8-16× speedup with only a slight reduction in accuracy (e.g., 1.0% F1 drop on ACE05), which makes our model fast and accurate to use in practice. Our final system is called PURE (the **P**rinceton **U**niversity **R**elation **E**xtraction system) and we make our code and models publicly available for the research community.

We summarize our contributions as follows:

- We present a simple and effective approach for end-to-end relation extraction, which learns two independent encoders for entity recognition and relation extraction. Our model establishes the new state-of-the-art on three standard benchmarks and surpasses all previous joint models.
- We conduct careful analyses to understand why our approach performs so well and how different factors impact the final performance. We conclude that it is more effective to learn distinct contextual representations for entities and relations than to learn them jointly.
- To speed up the inference time of our model, we also propose a novel efficient approximation, which achieves a large runtime improvement with only a small accuracy drop.

2 Related Work

Traditionally, extracting relations between entities in text has been studied as two separate tasks: named entity recognition and relation extraction. In the last several years, there has been a surge of interest in developing models for joint extraction of entities and relations (Li and Ji, 2014; Miwa and Sasaki, 2014; Miwa and Bansal, 2016). We group existing joint models into two categories: *structured prediction* and *multi-task learning*:

Structured prediction Structured prediction approaches cast the two tasks into one unified framework, although it can be formulated in various ways. Li and Ji (2014) propose an action-based system which identifies new entities as well as links to previous entities, Zhang et al. (2017a); Wang and Lu (2020) adopt a table-filling approach proposed in (Miwa and Sasaki, 2014); Katiyar and Cardie (2017) and Zheng et al. (2017) employ sequence tagging-based approaches; Sun et al. (2019) and Fu et al. (2019) propose graph-based approaches to jointly predict entity and relation types; and, Li et al. (2019) convert the task into a multi-turn question answering problem. All of these approaches need to tackle a global optimization problem and perform joint decoding at inference time, using beam search or reinforcement learning.

Multi-task learning This family of models essentially builds two separate models for entity

recognition and relation extraction and optimizes them together through parameter sharing. Miwa and Bansal (2016) propose to use a sequence tagging model for entity prediction and a tree-based LSTM model for relation extraction. The two models share one LSTM layer for contextualized word representations and they find sharing parameters improves performance (slightly) for both models. The approach of Bekoulis et al. (2018) is similar except that they model relation classification as a multi-label head selection problem. Note that these approaches still perform pipelined decoding: entities are first extracted and the relation model is applied on the predicted entities.

The closest work to ours is DYGIE and DYGIE++ (Luan et al., 2019; Wadden et al., 2019), which builds on recent span-based models for coreference resolution (Lee et al., 2017) and semantic role labeling (He et al., 2018). The key idea of their approaches is to learn shared span representations between the two tasks and update span representations through dynamic graph propagation layers. A more recent work Lin et al. (2020) further extends DYGIE++ by incorporating global features based on cross-substask and cross-instance constraints.[2] Our approach is much simpler and we will detail the differences in Section 3.2 and explain why our model performs better.

3 Method

In this section, we first formally define the problem of end-to-end relation extraction in Section 3.1 and then detail our approach in Section 3.2. Finally, we present our approximation solution in Section 3.3, which considerably improves the efficiency of our approach during inference.

3.1 Problem Definition

The input of the problem is a sentence X consisting of n tokens $x_1, x_2, \ldots, x_n$. Let $S = \{s_1, s_2, \ldots, s_m\}$ be all the possible spans in X of up to length L and $\text{START}(i)$ and $\text{END}(i)$ denote start and end indices of s_i. Optionally, we can incorporate cross-sentence context to build better contextual representations (Section 3.2). The problem can be decomposed into two sub-tasks:

Named entity recognition Let $\mathcal{E}$ denote a set of pre-defined entity types. The named entity recognition task is, for each span $s_i \in S$, to predict an entity type $y_e(s_i) \in \mathcal{E}$ or $y_e(s_i) = \epsilon$ representing span s_i is not an entity. The output of the task is $Y_e = \{(s_i, e) : s_i \in S, e \in \mathcal{E}\}$.

Relation extraction Let $\mathcal{R}$ denote a set of pre-defined relation types. The task is, for every pair of spans $s_i \in S, s_j \in S$, to predict a relation type $y_r(s_i, s_j) \in \mathcal{R}$, or there is no relation between them: $y_r(s_i, s_j) = \epsilon$. The output of the task is $Y_r = \{(s_i, s_j, r) : s_i, s_j \in S, r \in \mathcal{R}\}$.

3.2 Our Approach

As shown in Figure 1, our approach consists of an entity model and a relation model. The entity model first takes the input sentence and predicts an entity type (or ϵ) for each single span. We then process every pair of candidate entities independently in the relation model by inserting extra marker tokens to highlight the subject and object and their types. We will detail each component below, and finally summarize the differences between our approach and DYGIE++ (Wadden et al., 2019).

Entity model Our entity model is a standard span-based model following prior work (Lee et al., 2017; Luan et al., 2018, 2019; Wadden et al., 2019). We first use a pre-trained language model (e.g., BERT) to obtain contextualized representations $\mathbf{x}_t$ for each input token x_t. Given a span $s_i \in S$, the span representation $\mathbf{h}_e(s_i)$ is defined as:

$$\mathbf{h}_e(s_i) = [\mathbf{x}_{\text{START}(i)}; \mathbf{x}_{\text{END}(i)}; \phi(s_i)],$$

where $\phi(s_i) \in \mathbb{R}^{d_F}$ represents the learned embeddings of span width features. The span representation $\mathbf{h}_e(s_i)$ is then fed into a feedforward network to predict the probability distribution of the entity type $e \in \mathcal{E} \cup \{\epsilon\}$: $P_e(e \mid s_i)$.

Relation model The relation model aims to take a pair of spans s_i, s_j (a subject and an object) as input and predicts a relation type or ϵ. Previous approaches (Luan et al., 2018, 2019; Wadden et al., 2019) re-use the span representations $\mathbf{h}_e(s_i), \mathbf{h}_e(s_j)$ to predict the relationship between s_i and s_j. We hypothesize that these representations only capture contextual information around each individual entity and might fail to capture the dependencies between the pair of spans. We also argue that sharing the contextual representations between *different* pairs of spans may be suboptimal. For instance, the words *is a* in Figure 1 are crucial in understanding the relationship between MORPA and PARSER but not for MORPA and TEXT-TO-SPEECH.

[2] This is an orthogonal contribution to ours and we will explore it for future work.

Our relation model instead processes each pair of spans independently and inserts typed markers at the input layer to highlight the subject and object and their types. Specifically, given an input sentence X and a pair of subject-object spans s_i, s_j, where s_i, s_j have a type of $e_i, e_j \in \mathcal{E} \cup \{\epsilon\}$ respectively. We define text markers as $\langle \text{S}{:}e_i \rangle$, $\langle /\text{S}{:}e_i \rangle$, $\langle \text{O}{:}e_j \rangle$, and $\langle /\text{O}{:}e_j \rangle$, and insert them into the input sentence before and after the subject and object spans (Figure 1 (b)).[3] Let $\widehat{X}$ denote this modified sequence with text markers inserted:

this way model already knows entities (cant guess those wrong here)

$$\widehat{X} = \ldots \langle \text{S}{:}e_i \rangle, x_{\text{START}(i)}, \ldots, x_{\text{END}(i)}, \langle /\text{S}{:}e_i \rangle, \ldots \langle \text{O}{:}e_j \rangle, x_{\text{START}(j)}, \ldots, x_{\text{END}(j)}, \langle /\text{O}{:}e_j \rangle, \ldots .$$

We apply a *second* pre-trained encoder on $\widehat{X}$ and denote the output representations by $\widehat{\mathbf{x}}_t$. We concatenate the output representations of two start positions and obtain the span-pair representation:

$$\mathbf{h}_r(s_i, s_j) = [\widehat{\mathbf{x}}_{\widehat{\text{START}}(i)}; \widehat{\mathbf{x}}_{\widehat{\text{START}}(j)}],$$

where $\widehat{\text{START}}(i)$ and $\widehat{\text{START}}(j)$ are the indices of $\langle \text{S}{:}e_i \rangle$ and $\langle \text{O}{:}e_j \rangle$ in $\widehat{X}$. Finally, the representation $\mathbf{h}_r(s_i, s_j)$ will be fed into a feedforward network to predict the probability distribution of the relation type $r \in \mathcal{R} \cup \{\epsilon\}$: $P_r(r|s_i, s_j)$.

This idea of using additional markers to highlight the subject and object is not entirely new as it has been studied recently in relation classification (Zhang et al., 2019; Soares et al., 2019; Peters et al., 2019). However, most relation classification tasks (e.g., TACRED (Zhang et al., 2017b)) only focus on a given pair of subject and object in an input sentence and its effectiveness has not been evaluated in the end-to-end setting in which we need to classify the relationships between multiple entity mentions. We observed a large improvement in our experiments (Section 5.1) and this strengthens our hypothesis that modeling the relationship between different entity pairs in one sentence require different contextual representations. Furthermore, Zhang et al. (2019); Soares et al. (2019) only consider untyped markers (e.g., $\langle \text{S} \rangle$, $\langle /\text{S} \rangle$) and previous end-to-end models (e.g., (Wadden et al., 2019)) only inject the entity type information into the relation model through auxiliary losses. We find that injecting type information at the input layer is very helpful in distinguishing entity types — for example, whether "Disney" refers to a *person* or an *organization*— before trying to understand the relations.

[3] Our final model indeed only considers $e_i, e_j \neq \epsilon$. We have explored strategies using spans which are predicted as ϵ for the relation model but didn't find improvement. See Section 5.3 for more discussion.

Cross-sentence context Cross-sentence information can be used to help predict entity types and relations, especially for pronominal mentions. Luan et al. (2019); Wadden et al. (2019) employ a propagation mechanism to incorporate cross-sentence context. Wadden et al. (2019) also add a 3-sentence context window which is shown to improve performance. We also evaluate the importance of leveraging cross-sentence context in our approach. As we expect that pre-trained language models to be able to capture long-range dependencies, we simply incorporate cross-sentence context by extending the sentence to a fixed window size W for both the entity and relation model. Specifically, given an input sentence with n words, we augment the input with $(W-n)/2$ words from the left context and right context respectively.

Training & inference For both entity model and relation model, we fine-tune the two pre-trained language models using task-specific losses. We use cross-entropy loss for both models:

$$\mathcal{L}_e = -\sum_{s_i \in S} \log P_e(e_i^* | s_i)$$

$$\mathcal{L}_r = -\sum_{s_i, s_j \in S_G, s_i \neq s_j} \log P_r(r_{i,j}^* \mid s_i, s_j),$$

where e_i^* represents the gold entity type of s_i and $r_{i,j}^*$ represents the gold relation type of span pair s_i, s_j in the training data. For training the relation model, we only consider the gold entities $S_G \subset S$ in the training set and use the gold entity labels as the input of the relation model. We considered training on predicted entities as well as all spans S (with pruning), but none of them led to meaningful improvements compared to this simple pipelined training (see more discussion in Section 5.3). During inference, we first predict the entities by taking $y_e(s_i) = \arg\max_{e \in \mathcal{E} \cup \{\epsilon\}} P_e(e|s_i)$. Denote $S_{\text{pred}} = \{s_i : y_e(s_i) \neq \epsilon\}$, we enumerate all the spans $s_i, s_j \in S_{\text{pred}}$ and use $y_e(s_i), y_e(s_j)$ to construct the input for the relation model $P_r(r \mid s_i, s_j)$.

Differences from DYGIE++ Our approach differs from DYGIE++ (Luan et al., 2019; Wadden et al., 2019) in the following ways: (1) We use separate encoders for the entity and relation models, without any multi-task learning. The predicted entity types are used directly to construct the input for the relation model. (2) The contextual repre-

sentations in the relation model are specific to each pair of spans by using the text markers. (3) We only incorporate cross-sentence information by extending the input with additional context (as they did) and we do not employ any graph propagation layers and beam search.[4] As a result, our model is much simpler. As we will show in the experiments (Section 4), it also achieves large gains in all the benchmarks, using the same pre-trained encoders.

3.3 Efficient Batch Computations

One possible shortcoming of our approach is that we need to run our relation model once for every pair of entities. To alleviate this issue, we propose a novel and efficient alternative to our relation model. The key problem is that we would like to re-use computations for different pairs of spans in the same sentence. This is impossible in our original model because we must insert the entity markers for each pair of spans independently. To this end, we propose an approximation model by making two major changes to the original relation model. First, instead of directly inserting entity markers into the original sentence, we tie the position embeddings of the markers with the start and end tokens of the corresponding span:

$$\text{P}(\langle \text{S}{:}e_i \rangle), \text{P}(\langle /\text{S}{:}e_i \rangle) := \text{P}(x_{\text{START}(i)}), \text{P}(x_{\text{END}(i)})$$
$$\text{P}(\langle \text{O}{:}e_j \rangle), \text{P}(\langle /\text{O}{:}e_j \rangle) := \text{P}(x_{\text{START}(j)}), \text{P}(x_{\text{END}(j)}),$$

where $\text{P}(\cdot)$ denotes the position id of a token. As the example shown in Figure 1, if we want to classify the relationship between MORPA and PARSER, the first entity marker $\langle S{:}\ \text{METHOD} \rangle$ will share the position embedding with the token MOR. By doing this, the position embeddings of the original tokens will not be changed.

Second, we add a constraint to the attention layers. We enforce the text tokens to only attend to text tokens and not attend to the marker tokens while an entity marker token can attend to all the text tokens and all the 4 marker tokens associated with the same span pair. These two modifications allow us to re-use the computations of all text tokens, because the representations of text tokens are independent of the entity marker tokens. Thus, we can batch multiple pairs of spans from the same sentence in one run of the relation model. In practice, we add all marker tokens to the end of the sentence to form an input that batches a set of span pairs (Figure 1(c)). This leads to a large speedup at inference time and only a small drop in performance (Section 4.3).

[4] They also incorporated coreferences and event prediction in their framework. We focus on entity and relation extraction in this paper and we leave these extensions to future work.

4 Experiments

4.1 Setup

Datasets We evaluate our approach on three popular end-to-end relation extraction datasets: ACE05[5], ACE04[6], and SciERC (Luan et al., 2018). Table 2 shows the data statistics of each dataset. The ACE05 and ACE04 datasets are collected from a variety of domains, such as newswire and online forums. The SciERC dataset is collected from 500 AI paper abstracts and defines scientific terms and relations specially for scientific knowledge graph construction. We follow previous work and use the same preprocessing procedure and splits for all datasets. See Appendix A for more details.

Evaluation metrics We follow the standard evaluation protocol and use micro F1 measure as the evaluation metric. For named entity recognition, a predicted entity is considered as a correct prediction if its span boundaries and the predicted entity type are both correct. For relation extraction, we adopt two evaluation metrics: (1) *boundaries* evaluation (Rel): a predicted relation is considered as a correct prediction if the boundaries of two spans are correct and the predicted relation type is correct; (2) *strict* evaluation (Rel+): in addition to what is required in the *boundaries* evaluation, predicted entity types also must be correct. More discussion of the evaluation settings can be found in Bekoulis et al. (2018); Taillé et al. (2020).

Implementation details We use *bert-base-uncased* (Devlin et al., 2019) and *albert-xxlarge-v1* (Lan et al., 2020) as the base encoders for ACE04 and ACE05, for a fair comparison with previous work and an investigation of small vs large pre-trained models.[7] We also use *scibert-scivocab-uncased* (Beltagy et al., 2019) as the base encoder for SciERC, as this in-domain pre-trained model is shown to be more effective than BERT (Wadden et al., 2019). We use a context window size of $W = 300$ for the entity model and $W = 100$ for

[5] catalog.ldc.upenn.edu/LDC2006T06

[6] catalog.ldc.upenn.edu/LDC2005T09

[7] As detailed in Table 1, some previous work used BERT-large models. We are not able to do a comprehensive study of all the pre-trained models and our BERT-base results are generally higher than most published results using larger models.

Model	Encoder	ACE05			ACE04			SciERC		
		Ent	Rel	Rel+	Ent	Rel	Rel+	Ent	Rel	Rel+
(Li and Ji, 2014)	-	80.8	52.1	49.5	79.7	48.3	45.3	-	-	-
(Miwa and Bansal, 2016)	L	83.4	-	55.6	81.8	-	48.4	-	-	-
(Katiyar and Cardie, 2017)	L	82.6	55.9	53.6	79.6	49.3	45.7	-	-	-
(Zhang et al., 2017a)	L	83.6	-	57.5	-	-	-	-	-	-
(Luan et al., 2018)♣†	L+E	-	-	-	-	-	-	64.2	39.3	-
(Luan et al., 2019)♣†	L+E	88.4	63.2	-	87.4	59.7	-	65.2	41.6	-
(Li et al., 2019)	Bl	84.8	-	60.2	83.6	-	49.4	-	-	-
(Dixit and Al-Onaizan, 2019)	L+E	86.0	-	62.8	-	-	-	-	-	-
(Wadden et al., 2019)♣†	Bb	88.6	63.4	-	-	-	-	-	-	
(Wadden et al., 2019)♣†	SciB	-	-	-	-	-	-	67.5	48.4	
(Lin et al., 2020)	Bl	88.8	67.5	-	-	-	-	-	-	-
(Wang and Lu, 2020)	ALB	89.5	67.6	64.3	88.6	63.3	59.6	-	-	-
PURE (ours): single-sentence	Bb	88.7	66.7	63.9	88.1	62.8	58.3	-	-	-
	SciB	-	-	-	-	-	-	66.6	48.2	35.6
	ALB	89.7	69.0	65.6	88.8	64.7	60.2	-	-	-
PURE (ours): cross-sentence♣	Bb	90.1	67.7	64.8	89.2	63.9	60.1	-	-	-
	SciB	-	-	-	-	-	-	**68.9**	**50.1**	**36.8**
	ALB	**90.9**	**69.4**	**67.0**	**90.3**	**66.1**	**62.2**	-	-	-

Table 1: Test F1 scores on ACE04, ACE05, and SciERC. We evaluate our approach in two settings: *single-sentence* and *cross-sentence* depending on whether cross-sentence context is used or not. ♣: These models leverage cross-sentence information. †: These models are trained with additional data (e.g., coreference). The encoders used in different models: L = LSTM, L+E = LSTM + ELMo, Bb = BERT-base, Bl = BERT-large, SciB = SciBERT (size as BERT-base), ALB = ALBERT-xxlarge-v1. *Rel* denotes the *boundaries* evaluation (the entity boundaries must be correct) and *Rel+* denotes the *strict* evaluation (both the entity boundaries and types must be correct).

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	# Sentences		
			Train	Dev	Test
ACE05	7	6	10,051	2,424	2,050
ACE04	7	6	8,683 (5-fold)		
SciERC	6	7	1,861	275	551

Table 2: The statistics of the datasets. We use ACE04, ACE05, and SciERC for evaluating end-to-end relation extraction.

the relation model in our default setting using cross-sentence context[8] and the effect of different context sizes is provided in Section 5.4. We consider spans up to $L = 8$ words. For all the experiments, we report the averaged F1 scores of 5 runs. More implementation details can be found in Appendix B.

4.2 Main Results

Table 1 compares our approach PURE to all the previous results. We report the F1 scores in both single-sentence and cross-sentence settings. As is shown, our single-sentence models achieve strong performance and incorporating cross-sentence context further improves the results considerably. Our BERT-base (or SciBERT) models achieve similar or better results compared to all the previous work including models built on top of larger pre-trained LMs, and our results are further improved by using a larger encoder ALBERT.

For entity recognition, our best model achieves an absolute F1 improvement of $+1.4\%$, $+1.7\%$, $+1.4\%$ on ACE05, ACE04, and SciERC respectively. This shows that cross-sentence information is useful for the entity model and pre-trained Transformer encoders are able to capture long-range dependencies from a large context. For relation extraction, our approach outperforms the best previous methods by an absolute F1 of $+1.8\%$, $+2.8\%$, $+1.7\%$ on ACE05, ACE04, and SciERC respectively. We also obtained a 4.3% higher relation F1 on ACE05 compared to DYGIE++ (Wadden et al., 2019) using the same BERT-base pre-trained model. Compared to the previous best approaches using either global features (Lin et al., 2020) or complex neural models (e.g., MT-RNNs) (Wang and Lu, 2020), our approach is much simpler and achieves large improvements on all the datasets. Such improvements demonstrate the effectiveness

[8] We use a context window size $W = 100$ for the ALBERT entity models to reduce GPU memory usage.

Model	ACE05		SciERC	
	Rel (F1)	Speed (sent/s)	Rel (F1)	Speed (sent/s)
Full (single)	**66.7**	32.1	**48.2**	34.6
Approx. (single)	65.7	**384.7**	47.0	**301.1**
Full (cross)	**67.7**	14.7	**50.1**	19.9
Approx. (cross)	66.5	**237.6**	48.8	**194.7**

Table 3: We compare our full relation model and the approximation model in both accuracy and speed. The accuracy is measured as the relation F1 (boundaries) on the test set. These results are obtained using BERT-base for ACE05 and SciBERT for SciERC in both single-sentence and cross-sentence settings. The speed is measured on a single NVIDIA GeForce 2080 Ti GPU with a batch size of 32.

of learning representations for entities and relations of different entity pairs, as well as early fusion of entity information in the relation model. We also noticed that compared to the previous state-of-the-art model (Wang and Lu, 2020) based on ALBERT, our model achieves a similar entity F1 (89.5 vs 89.7) but a substantially better relation F1 (67.6 vs 69.0) without using context. This clearly demonstrates the superiority of our relation model. Finally, we also compare our model to a joint model (similar to DYGIE++) of different data sizes to test the generality of our results. As shown in Appendix C, our findings are robust to data sizes.

4.3 Batch Computations and Speedup

In Section 3.3, we proposed an efficient approximation solution for the relation model, which enables us to re-use the computations of text tokens and batch multiple span pairs in one input sentence. We evaluate this approximation model on ACE05 and SciERC. Table 3 shows the relation F1 scores and the inference speed of the full relation model and the approximation model. On both datasets, our approximation model significantly improves the efficiency of the inference process.[9] For example, we obtain a $11.9\times$ speedup on ACE05 and a $8.7\times$ speedup on SciERC in the single-sentence setting. By re-using a large part of computations, we are able to make predictions on the full ACE05 test set (2k sentences) in less than 10 seconds on a single GPU. On the other hand, this approximation only leads to a small performance drop and the relaion F1 measure decreases by only 1.0% and 1.2% on ACE05 and SciERC respectively in the single-sentence setting. Considering the accuracy and efficiency of this approximation model, we expect it to be very effective to use in practice.

[9]Note that we only applied this batch computation trick at inference time, because we observed that training with batch computation leads to a slightly (and consistently) worse result. We hypothesize that this is due to the impact of increased batch sizes. We still modified the position embedding and attention masks during training (without batching the instances though).

5 Analysis

Despite its simple design and training paradigm, we have shown that our approach outperforms all previous joint models. In this section, we aim to take a deeper look and understand what contributes to its final performance.

5.1 Importance of Typed Text Markers

Our key observation is that it is crucial to build different contextual representations for different pairs of spans and an early fusion of entity type information can further improve performance. To validate this, we experiment the following variants on both ACE05 and SciERC:

TEXT: We use the span representations defined in the entity model (Section 3.2) and concatenate the hidden representations for the subject and the object, as well as their element-wise multiplication: $[\mathbf{h}_e(s_i), \mathbf{h}_e(s_j), \mathbf{h}_e(s_i) \odot \mathbf{h}_e(s_j)]$. This is similar to the relation model in Luan et al. (2018, 2019).

TEXTETYPE: We concatenate the span-pair representations from TEXT with entity type embeddings $\psi(e_i), \psi(e_j) \in \mathbb{R}^{d_E}$ (d_E = 150).

MARKERS: We use untyped entity types ($\langle$S$\rangle$, $\langle$/S$\rangle$, $\langle$O$\rangle$, $\langle$/O$\rangle$) at the input layer and concatenate the representations of two spans' starting points.

MARKERSETYPE: We concatenate the span-pair representations from MARKERS with entity type embeddings $\psi(e_i), \psi(e_j) \in \mathbb{R}^{d_E}$ (d_E = 150).

MARKERSELOSS: We also consider a variant which uses untyped markers but add another FFNN to predict the entity types of subject and object through auxiliary losses. This is similar to how the entity information is used in multi-task learning (Luan et al., 2019; Wadden et al., 2019).

TYPEDMARKERS: This is our final model described in Section 3.2 with typed entity markers.

Table 4 summarizes the results of all the variants using either *gold* entities or *predicted* entities from the entity model. As is shown, different input representations make a clear difference and the variants of using marker tokens are significantly

Input	ACE05		SciERC	
	gold	e2e	gold	e2e
TEXT	67.6	61.6	61.7	45.3
TEXTETYPE	68.2	62.6	63.6	45.7
MARKERS	70.5	63.3	68.2	49.1
MARKERSETYPE	71.3	63.8	68.9	**49.7**
MARKERSELOSS	70.7	63.6	68.0	49.2
TYPEDMARKERS	**72.6**	**64.2**	**69.1**	**49.7**

Table 4: Relation F1 (boundaries) on the development set of ACE05 and SciERC with different input features. *e2e*: the entities are predicted by our entity model; *gold*: the gold entities are given. The results are obtained using BERT-base with *single-sentence* context for ACE05 and SciBERT with *cross-sentence* context for SciERC. For both ACE05 and SciERC, we use the same entity models with *cross-sentence* context to compute the *e2e* scores of using different input features.

Shared encoder?	Enity F1	Relation F1
✗	**88.8**	**64.8**
✓	87.7	64.4

Table 5: Relation F1 (boundaries) scores when entity and relation encoders are shared and *not* shared on the ACE05 development set. This result is obtained from BERT-base models with cross-sentence context.

better than standard text representations and this suggests the importance of learning different representations with respect to different pairs of spans. Compared to TEXT, TYPEDMARKERS improved the F1 scores dramatically by $+5.0\%$ and $+7.4\%$ absolute when gold entities are given. With the predicted entities, the improvement is reduced as expected while it remains large enough. Finally, entity type is useful in improving the relation performance and an early fusion of entity information is particularly effective (TYPEDMARKERS vs MARKERSETYPE and MARKERSELOSS). We also find that MARKERSETYPE to perform even better than MARKERSELOSS which suggests that using entity types directly as features is better than using them to provide training signals through auxiliary losses.

5.2 Modeling Entity-Relation Interactions

One main argument for joint models is that modeling the interactions between the two tasks can contribute to each other. In this section, we aim to validate if it is the case in our approach. We first study whether sharing the two representation encoders can improve performance or not. We train the entity and relation models together by jointly optimizing $\mathcal{L}_e + \mathcal{L}_r$ (Table 5). We find that simply sharing the encoders hurts both the entity and relation F1. We think this is because the two tasks have different input formats and require different features for predicting entity types and relations, thus using separate encoders indeed learns better task-specific features. We also explore whether the relation information can improve the entity performance. To do so, we add an auxiliary loss to our entity model, which concatenates the two span representations as well as their element-wise multiplication (see the TEXT variant in Section 5.1) and predicts the relation type between the two spans ($r \in \mathcal{R}$ or ϵ). Through joint training with this auxiliary relation loss, we observe a negligible improvement ($< 0.1\%$) on averaged entity F1 over 5 runs on the ACE05 development set. To summarize, (1) entity information is clearly important in predicting relations (Section 5.1). However, we don't find that relation information to improve our entity model substantially[10]; (2) simply sharing the encoders does not provide benefits to our approach.

	ACE05	SciERC
Gold entities	**64.8**	49.7
10-way jackknifing	63.9	48.1
$0.4n$ spans (typed)	64.6	**50.2**
$0.4n$ spans (untyped)	56.9	48.4
$0.4n$ spans (untyped + eloss)	63.0	48.5

Table 6: We compare relation F1 (boundaries) with different training strategies on the development sets of ACE05 and SciERC. This result is from training BERT-base and SciBERT models with cross-sentence context. *typed*: typed markers, *untyped*: untyped markers, *untyped + eloss*: untyped markers with auxiliary entity loss. See text for more details.

5.3 Mitigating Error Propagation

A well-known drawback of pipeline training is the error propagation issue. In our final model, we use gold entities (and their types) to train the relation model and the predicted entities during inference and this may lead to a discrepancy between training and testing. In the following, we describe several attempts we made to address this issue.

We first study whether using predicted entities

[10] Miwa and Bansal (2016) observed a slight improvement on entity F1 by sharing the parameters (80.8 → 81.8 F1) on the ACE05 development data. Wadden et al. (2019) observed that their relation propagation layers improved the entity F1 slightly on SciERC but it hurts performance on ACE05.

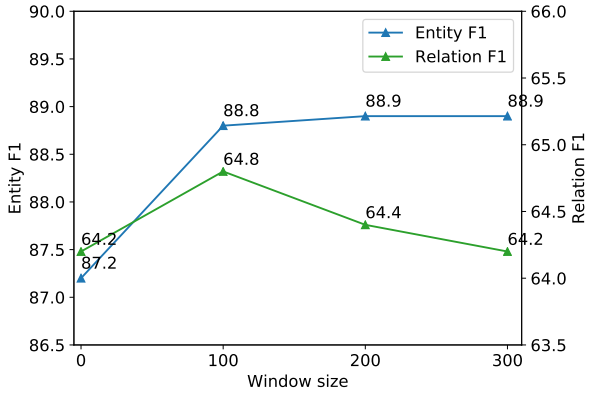

Figure 2: Effect of different context window sizes, measured on the ACE05 development set with the BERT-base model. We use the same entity model (an entity model with $W = 300$) to report the relation F1 scores (boundaries).

— instead of gold entities — during training can mitigate this issue. We adopt a 10-way jackknifing method, which is a standard technique in many NLP tasks such as dependency parsing (Agić and Schluter, 2017). Specifically, we divide the data into 10 folds and predict the entities in the k-th fold using an entity model trained on the remainder. As shown in Table 6, we find that jackknifing strategy hurts the final relation performance surprisingly. We hypothesize that it is because it introduced additional noise during training.

Second, we consider using more pairs of spans for the relation model at both training and testing time. The main reason is that in the current pipeline approach, if a gold entity is missed out by the entity model during inference, the relation model will not be able to predict any relations associated with that entity. Following the beam search strategy used in the previous work (Luan et al., 2019; Wadden et al., 2019), we consider using λn ($\lambda = 0.4$ and n is the sentence length)[11] top spans scored by the entity model. We explored several different strategies for encoding the top-scoring spans for the relation model: (1) typed markers: the same as our main model except that we now have markers e.g., $\langle \text{S}{:}\epsilon \rangle$, $\langle /\text{S}{:}\epsilon \rangle$ as input tokens; (2) untyped markers: in this case, the relation model is unaware of a span is an entity or not; (3) untyped markers trained with an auxiliary entity loss ($e \in \mathcal{E}$ or ϵ). As Table 6 shows, none of these changes led to significant improvements and using untyped markers is especially worse because the relation model struggles to identify whether a span is an entity or not.

In sum, we do not find any of these attempts improved performance significantly and our simple pipelined training turns out to be a surprisingly effective strategy. We do not argue that this error propagation issue does not exist or cannot be solved, while we will need to explore better solutions to address this issue.

5.4 Effect of Cross-sentence Context

In Table 1, we demonstrated the improvements from using cross-sentence context on both the entity and relation performance. We explore the effect of different context sizes W in Figure 2. We find that using cross-sentence context clearly improves both entity and relation F1. However, we find the relation performance doesn not further increase from $W = 100$ to $W = 300$. In our final models, we use $W = 300$ for the entity model and $W = 100$ for the relation model.

6 Conclusion

In this paper, we present a simple and effective approach for end-to-end relation extraction. Our model learns two encoders for entity recognition and relation extraction independently and our experiments show that it outperforms previous state-of-the-art on three standard benchmarks considerably. We conduct extensive analyses to undertand the superior performance of our approach and validate the importance of learning distinct contextual representations for entities and relations and using entity information as input features for the relation model. We also propose an efficient approximation, obtaining a large speedup at inference time with a small reduction in accuracy. We hope that this simple model will serve as a very strong baseline and make us rethink the value of joint training in end-to-end relation extraction.

Acknowledgements

We thank Yi Luan for the help with the datasets and evaluation. We thank Howard Chen, Ameet Deshpande, Dan Friedman, Karthik Narasimhan, and the anonymous reviewers for their helpful comments and feedback. This work is supported in part by a Graduate Fellowship at Princeton University.

[11] This pruning strategy achieves a recall of 96.7% of gold relations on the development set of ACE05.

References

Željko Agić and Natalie Schluter. 2017. How (not) to train a dependency parser: The curious case of jackknifing part-of-speech taggers. In *Association for Computational Linguistics (ACL)*, pages 679–684.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Adversarial training for multi-context joint entity and relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2830–2836.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 3606–3611.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 724–731.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 551–560.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186.

Kalpit Dixit and Yaser Al-Onaizan. 2019. Span-level model for relation extraction. In *Association for Computational Linguistics (ACL)*, pages 5308–5314.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, H Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1–8.

Radu Florian, Hongyan Jing, Nanda Kambhatla, and Imed Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *Association for Computational Linguistics (ACL)*, pages 473–480.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Association for Computational Linguistics (ACL)*, pages 1409–1418.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Association for Computational Linguistics (ACL)*, pages 364–369.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Association for Computational Linguistics (ACL)*, pages 178–181.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Association for Computational Linguistics (ACL)*, pages 917–928.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 188–197.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Association for Computational Linguistics (ACL)*, pages 402–412.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *Association for Computational Linguistics (ACL)*, pages 1340–1350.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Association for Computational Linguistics (ACL)*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 3219–3232.

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

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Association for Computational Linguistics (ACL)*, pages 1105–1116.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869.

Matthew E Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 43–54.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Computational Natural Language Learning (CoNLL)*, pages 147–155.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Computational Natural Language Learning (CoNLL)*, pages 142–147.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Association for Computational Linguistics (ACL)*, pages 2895–2905.

Changzhi Sun, Yeyun Gong, Yuanbin Wu, Ming Gong, Daxin Jiang, Man Lan, Shiliang Sun, and Nan Duan. 2019. Joint type inference on entities and relations via graph convolutional networks. In *Association for Computational Linguistics (ACL)*, pages 1361–1370.

Bruno Taillé, Vincent Guigue, Geoffrey Scoutheeten, and Patrick Gallinari. 2020. Let's stop incorrect comparisons in end-to-end relation extraction! In *Empirical Methods in Natural Language Processing (EMNLP)*.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 5788–5793.

Jue Wang and Wei Lu. 2020. Two are better than one: Joint entity and relation extraction with table-sequence encoders. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 71–78.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017a. End-to-end neural relation extraction with global optimization. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1730–1740.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017b. Position-aware attention and supervised data improve slot filling. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 35–45.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Association for Computational Linguistics (ACL)*, pages 1441–1451.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Association for Computational Linguistics (ACL)*, pages 1227–1236.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Association for Computational Linguistics (ACL)*, pages 427–434.

A Datasets

We use ACE04, ACE05, and SciERC datasets in our experiments. Table 2 shows the data statistics of each dataset.

The ACE04 and ACE05 datasets are collected from a variety of domains, such as newswire and online forums. We follow Luan et al. (2019)'s preprocessing steps[12] and split ACE04 into 5 folds and ACE05 into train, development, and test sets.

The SciERC dataset is collected from 12 AI conference/workshop proceedings in four AI communities (Luan et al., 2018). SciERC includes annotations for scientific entities, their relations, and coreference clusters. We ignore the coreference annotations in our experiments. We use the processed dataset which is downloaded from the project website[13] of Luan et al. (2018).

B Implementation Details

We implement our models based on HuggingFace's *Transformers* library (Wolf et al., 2019). For the entity model, we follow Wadden et al. (2019) and set the width embedding size as $d_F = 150$ and use a 2-layer FFNN with 150 hidden units and ReLU activations to predict the probability distribution of entity types:

$$P_e(e \mid s_i) = \text{softmax}(\mathbf{W}_e\text{FFNN}(\mathbf{h}_e(s_i)).$$

For the relation model, we use a linear classifier on top of the span pair representation to predict the probability distribution of relation types:

$$P_r(r|s_i, s_j) = \text{softmax}(\mathbf{W}_r\mathbf{h}_r(s_i, s_j)).$$

For our approximation model (Section 4.3), we batch candidate pairs by adding 4 markers for each pair to the end of the sentence, until the total number of tokens exceeds 250. We train our models with Adam optimizer of a linear scheduler with a warmup ratio of 0.1. For all the experiments, we train the entity model for 100 epochs, and a learning rate of 1e-5 for weights in pre-trained LMs, 5e-4 for others and a batch size of 16. We train the relation model for 10 epochs with a learning rate of 2e-5 and a batch size of 32.

C Performance with Varying Data Sizes

We compare our pipeline model to a joint model with 10%, 25%, 50%, 100% of training data on the ACE05 dataset. Here, our goal is to understand whether our finding still holds when the training data is smaller (and hence it is expected to have more errors in entity predictions).

Training data	Ours		Joint	
	Ent	Rel	Ent	Rel
10%	**82.0**	**46.9**	81.5	37.0
25%	**84.9**	**57.6**	84.6	49.0
50%	85.5	**61.9**	**86.2**	57.7
100%	87.2	**63.4**	**87.4**	61.0

Table 7: F1 scores on ACE05 development set when only a subset of training samples (10%, 25%, 50%, or 100%) are provided.

Our baseline of joint model is our reimplementation of DYGIE++ (Wadden et al., 2019), without using propagation layers (the encoders are shared for the entity and relation model and no input marker is used; the top scoring $0.4n$ entities are considered in beam pruning). As shown in Table 7, we find that our model achieves even larger gains in relation F1 over the joint model, when the number of training examples is reduced. This further highlights the importance of explicitly encoding entity boundaries and type features in data-scarce scenarios.

[12] We use the script provided by Luan et al. (2019): https://github.com/luanyi/DyGIE/tree/master/preprocessing.

[13] http://nlp.cs.washington.edu/sciIE/