An Empirical Study of Pipeline vs. Joint Approaches to Entity and Relation Extraction

Zhaohui Yan*, Zixia Jia* and Kewei Tu
School of Information Science and Technology, ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
{yanzhh, jiazx, tukw}@shanghaitech.edu.cn *

Abstract

The Entity and Relation Extraction (ERE) task includes two basic sub-tasks: Named Entity Recognition and Relation Extraction. In the last several years, much work focused on joint approaches for the common perception that the pipeline approach suffers from the error propagation problem. Recent work reconsiders the pipeline scheme and shows that it can produce comparable results. To systematically study the pros and cons of these two schemes. We design and test eight pipeline and joint approaches to the ERE task. We find that with the same span representation methods, the best joint approach still outperforms the best pipeline model, but improperly designed joint approaches may have poor performance. We hope our work could shed some light on the pipeline-vs-joint debate of the ERE task and inspire further research.[1]

1 Introduction

The Entity and Relation Extraction (ERE) task aims to extract entities and their relations from unstructured text and is a fundamental task in the area of information extraction. There are two typical approaches to the ERE task: one is the pipeline approach (Chan and Roth, 2011) consisting of two models for the two sub-tasks, Named Entity Recognition (NER) and Relation Extraction (RE), respectively. Another is the joint approach that models the two sub-tasks jointly (Miwa and Sasaki, 2014; Zheng et al., 2017; Wang and Lu, 2020; Eberts and Ulges, 2020).

Pipeline approaches do not share any parameters between sub-tasks and decode sequentially. For joint approaches, one typical method is to share encoders across sub-tasks and performs pipelined decoding (Miwa and Bansal, 2016). Another method uses joint inference in addition to shared encoders, for example, Wang and Lu (2020) cast the ERE task into a table-filling problem. Among these joint approaches, some span-based joint approaches (Sun et al., 2019) have different task-dividing strategy. Span-based models embed each span for an input sentence and there are $O(n^4)$ possible span pairs. To reduce the high complexity, previous span-based joint approaches pre-identify entity spans and then use a cross-task module for the entity and relation type deduction. To clarify the definitions, we define a purely joint approach as a method with only a cross-task module for sub-tasks, a purely pipelined approach has no cross-task module.

It is generally believed that the pipeline approach suffers from the problem of error propagation, while the joint approach could leverage interactions between sub-tasks. However, recent research from Zhong and Chen (2021) shows that the *feature confusion* problem of the joint model may negate its benefits.There is also some work (Yan et al., 2021) that disagree with their conclusion and propose a new state-of-the-art approach. However, these studies are based on different settings and hence cannot be directly compared.

The debate on pipeline vs. joint approaches motivates us to perform a systematically empirical study. For a fair comparison, we design pipeline and joint approaches with similar settings. the pipeline works recently (Zhong and Chen, 2021; Ye et al., 2022) use span-based models to better leverage span-level features, we also adopt this setting. For previous span-based joint approaches (Sun et al., 2019) divide the NER task into two sub-tasks, this leads to a second level of pipeline vs. joint dilemma which we also wish to investigate. Specifically, we consider four sub-tasks for ERE: entity identification (Eid), entity classification (Ecls), relation identification (Rid) and relation classification (Rcls). We design ten modules for these sub-tasks and connect them to build eight

* Authors with symbol * have equal contributions and Kewei Tu is the corresponding auther.

[1]Source code is availabel at https://github.com/yanzhh/JointERE.

Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 437–443
November 20–23, 2022.

Module No.		Ei 1	Ec 2	Ri 3	Rc 4	Ner 5	Re 6	EcRi 7	NerRi 8	EcRe 9	NerRe 10
NER	Eid	•				•			•		•
	Ecls		•			•		•	•	•	•
RE	Rid			•			•	•	•	•	•
	Rcls				•		•			•	•

No.	Approaches	No.	Approaches
a1	Ei- Ec- Ri- Rc	a5	Ei- EcRi
a2	Ei- Ec- Re	a6	Ei- EcRe
a3	Ner- Ri- Rc	a7	NerRi- Rc
a4	Ner- Re	a8	NerRe

Table 1: The upper half is the list of ten modules. The dot in columns indicates the sub-tasks of a module. The lower half is the list of eight approaches

approaches as shown in Table 1. Following the recent work (Jia et al., 2022a), we use high-order inference to better exploit the correlation between sub-tasks. To experiment with the full joint span-based approach, we use span pruner in addition to an entity pre-identifier for our joint approaches. With the high recall of the span pruner, our last approach a8 can be viewed as a full joint approach.

Our empirical study shows that with the same embedding method, the pipeline approach could achieve competitive results even compared to some joint approaches, but the full joint approach still outperforms all the pipeline approaches.

2 Our methods

As mentioned above, there are ten modules solving different sub-task combinations and eight approaches. The former four approaches are fully pipelined, the final one is the fully joint approach and the others are approaches with cross-task modules. We first introduce all the modules and then describe the training and decoding processes of the approaches.

We denote an input sequence with n tokens as $X = \{x_1, x_2, ..., x_n\}$. m candidate spans of these tokens can be denoted as $S = \{s_i | 1 \leq i \leq m\}$, START$(i)$ and END(i) represent the head and tail token indices of s_i. The gold entity label set and the gold relation label set are represented as $\mathcal{E}$ and $\mathcal{R}$ respectively.

2.1 Encoding

For each module, we feed the token sequence into a pre-trained language model. For each token x_i, we use the embedding of the first sub-token from the last layer as the contextualized representations $\mathbf{x}_i$.

Follow Zhong and Chen (2021), for a given span $s_i \in S$, the span representation $\mathbf{b}_i$ is defined as:

$$\mathbf{b}_i = [\mathbf{x}_{\text{START}(i)}; \mathbf{x}_{\text{END}(i)}; \phi(s_i)]$$

where $\phi(s_i) \in R^{d_l}$ is a learned embedding of the span length. For each module, we feed the span representations into a two-layer MLP to get an R^{d_s}-dimension hidden vector $\mathbf{h}_i$ and for modules involving RE, we obtain span pair representations in a similar way:

$$\mathbf{h}_i = \text{MLP}_{\text{span}}(\mathbf{b}_i), \quad \mathbf{h}_{ij} = \text{MLP}_{\text{rel}}([\mathbf{b}_i; \mathbf{b}_j])$$

2.2 Single-task modules

For modules 1-6, the span or span pair representations are fed into a linear layer to score the span or span pair for each label.

$$\mathbf{g}_i = \text{Linear}_{\text{ent}}(\mathbf{h}_i), \quad \mathbf{g}_{ij} = \text{Linear}_{\text{rel}}(\mathbf{h}_{ij})$$

$\mathbf{g}_i \in R^{d_{\text{ent}}}$ and $\mathbf{g}_{ij} \in R^{d_{\text{rel}}}$. For classification modules Ec and Rc, $d_{\text{ent}} = |\mathcal{E}|$ and $d_{\text{rel}} = |\mathcal{R}|$. For module Ner involving both entity identification and classification, we add a *Null* label representing that the span is not an entity, so we have $d_{\text{ent}} = |\mathcal{E}| + 1$. Similarly, we have $d_{\text{rel}} = |\mathcal{R}| + 1$ for module Re. For the identification modules Ei and Ri, we set $d_{\text{ent}} = d_{\text{rel}} = 1$, meaning that we only score the existence of an entity or relation and fix the non-existence score to zero. The prediction of a span or a span pair is the label with the largest score among the gold label set, or we identify the span or relation with a score larger than 0.

2.3 Cross-task modules

Our cross-task modules adopt high-order inference (Jia et al., 2022b). There are two types of scores in the modules: unary scores and ternary scores.

The unary score of a span or a span pair captures the prior distribution information and is computed solely based on the feature of the variable. The unary score $\mathbf{g}_i$ for the span or $\mathbf{g}_{ij}$ for the span pair is the same as defined in single-task modules.

The ternary score is defined cover a span pair that captures the three-way correlation between their entity labels and the label of the relation between them. Specifically, for each span pair (s_i, s_j), we calculate a score tensor $\mathbf{f}_{ij} \in R^{(d_{\text{ent}})^2(d_{\text{rel}})}$ as follows. First, two separate linear transformations project the head and tail span representations into d_{t}-dimension hidden space:

$$\mathbf{h}_i^{\text{t}} = \text{Linear}_{\text{head}}^{\text{t}}(\mathbf{b}_i), \quad \mathbf{h}_j^{\text{t}} = \text{Linear}_{\text{tail}}^{\text{t}}(\mathbf{b}_j)$$

Then a weight tensor $W_{\text{t}} \in R^{d_{\text{t}} \times (d_{\text{ent}})^2 (d_{\text{rel}})}$ is used to transform the element-wise product $\mathbf{h}_i^{\text{t}} \circ \mathbf{h}_j^{\text{t}}$ into the score tensor $\mathbf{f}_{ij}$:

$$\mathbf{f}_{ij} = (\mathbf{h}_i^{\text{t}} \circ \mathbf{h}_j^{\text{t}}) W_{\text{t}}$$

High-order Inference The first-order inference is based solely on the unary score and the high-order inference is based on both the unary score and the tenary score. We follow Jia et al. (2022a) and the Mean-field Variational Inference (MVFI) for high-order inference which iteratively updates a factorized variational distribution Q to approximate the posterior label distribution. Specifically, $Q_i(e)$ represents the probability of span s_i having entity type e and $Q_{ij}(r)$ represents the probability of spans s_i, s_j having a relation of type r. For simplicity, we use $g_i(a), g_j(b), g_{ij}(r), f_{ij}(a,b,r)$ to represent the unary and ternary scores of spans s_i, s_j having entity types a, b and a relation of type r between them. Messages delivered for entity and relation types are updated as follows:

$$F_i^{\text{T}}(a) = \sum_j \sum_b Q_j^{\text{T}-1}(b) \sum_r \left(Q_{ij}^{\text{T}-1}(r) f_{ij}(a,b,r) + Q_{ji}^{\text{T}-1}(r) f_{ji}(b,a,r)\right)$$

$$F_{ij}^{\text{T}}(r) = \sum_{e_i} \sum_{e_j} Q_i^{\text{T}}(e_i) Q_j^{\text{T}-1}(e_j) f_{ij}(e_i, e_j, r)$$

The messages are then used to update the posterior distributions Q:

$$Q_i^{\text{T}}(e) \propto \exp(g_i(e) + F_i^{\text{T}}(e))$$
$$Q_{ij}^{\text{T}}(r) \propto \exp(g_{ij}(r) + F_{ij}^{\text{T}}(r))$$

With the distribution Q, we choose the label with the highest probability. For `EcRi` and `NerRi`, the $Q_{ij} > 0.5$ represents that the relation exists between the span pair (s_i, s_j).

2.4 Training and decoding

Training With the modules defined above, we build eight approaches as shown in Table 1. `a8` is an end-to-end joint model consisting of only `NerRe` and all the other approaches are pipelines of two or more modules. We train different modules in an approach independently without sharing any parameters. We train module `Ei` and `Ner` on all possible $O(nL)$ spans with a span length limit L. For `a7` and `a8`, we cannot train the cross-task modules `NerRi` and `NerRe` on all spans for the high complexity$O(n^2L^2|\mathcal{E}|^2|\mathcal{R}|)$, so we use a pre-trained pruner (see Appendix A for details) which identifies $O(n)$ most likely spans for both approaches and reduce the computational complexity to $O(n^2|\mathcal{E}|^2|\mathcal{R}|)$. For the downstream modules in `a1-a7`, we train them on the gold entity set or the span pair set built by enumerating all the spans s_i, s_j in the gold entity set following Zhong and Chen (2021). For example, with the span set $S = \{s_1, s_2, ..., s_m\}$, we build the span pair set $\{(s_1, s_2), ..., (s_1, s_m), ..., (s_i, s_{i+1}), ..., (s_i, s_m), ...\}$.

There are two loss functions for these modules:

$$L_{\text{ent}} = -\sum_{s_i \in S} \log P_i(e_i^*), L_{\text{rel}} = -\sum_{s_i, s_j \in S, i \neq j} \log P_{ij}(r_{ij}^*)$$

e_i^* and r_{ij}^* are the gold labels for span s_i and span pair (s_i, s_j) respectively. For cross-task modules, we have $P_i(e_i) = Q_i(e_i)$ and $P_{ij}(r_{ij}) = Q_{ij}(r_{ij})$; for the other modules, we have $P_i(e_i) = Softmax(g_i(e_i))$ and $P_{ij}(r_{ij}) = Softmax(g_{ij}(r_{ij}))$. The training objective of a module is to minimize $L = I_{\text{ent}} L_{\text{ent}} + I_{\text{rel}} L_{\text{rel}}$ where the $I_{\text{ent}}, I_{\text{rel}}$ indicate whether the module predicts entities and relations respectively. High-order inference with the MFVI is end-to-end differentiable.

Decoding For the pipeline approaches `a1-a7` , the decoding is a cascade process. The upstream module is decoded first and each downstream module builds the input using the output of the upstream module.

3 Experiments

3.1 Experimental settings

Datasets We experiment on two popular relation extraction datasets: ACE2005 (Christopher Walker and Maeda, 2006) and SciERC (Luan et al., 2018). We adopt the official training/validation/testing splits.

Evaluations We follow previous works and use the F1 scores with micro-averaging as the evaluation metric.

Specifically, for the NER task, a predicted entity is considered correctly identified (*Ent-I*) if its boundary matches the corresponding gold entity and correctly classified (*Ent-C*) if its type also matches. For RE tasks, the predicted relation is correctly identified (*Rel-I*) if the boundaries of its endpoints are correct and correctly classified (*Rel-C*) if the relation type matches the corresponding gold relation. To evaluate both tasks, the strict evaluation (*Rel*$^+$*-I* and *Rel*$^+$*-C*) requires correctly

	Approaches	ACE2005						SciERC					
		Ent-I	Ent-C	Rel-I	Rel-C	Rel$^+$-I	Rel$^+$-C	Ent-I	Ent-C	Rel-I	Rel-C	Rel$^+$-I	Rel$^+$-C
a1	Ei-Ec-Ri-Rc	94.41	88.29	71.52	66.56	66.53	62.75	79.17	67.06	51.47	47.03	38.10	35.39
a2	Ei-Ec-Re	94.41	88.29	**72.01**	67.30	67.30	63.66	79.17	67.06	51.41	47.52	38.04	35.56
a3	Ner-Ri-Rc	94.51	88.53	71.75	66.88	67.02	63.18	**79.44**	67.41	51.63	47.37	37.79	35.11
a4	Ner-Re	94.51	88.53	71.99	**67.50**	67.35	63.73	**79.44**	67.41	**52.43**	**48.68**	38.32	35.86
a5	Ei-EcRi-Rc	94.41	88.14	71.30	66.37	67.09	63.12	79.17	66.75	49.90	45.51	37.15	34.20
a6	Ei-EcRe	94.41	87.94	71.09	66.46	67.06	63.81	79.17	66.42	50.10	46.31	37.36	34.84
a7	NerRi-Rc	94.55	**88.60**	70.31	66.20	67.01	63.51	79.31	67.63	47.31	43.75	36.70	34.25
a8	NerRe	**94.57**	88.51	71.50	67.34	**67.71**	**64.66**	79.37	**68.01**	51.26	47.51	**40.05**	**37.42**

Table 2: F1 scores on ACE2005 and SciERC

Rel$^+$-C	ACE2005			SciERC		
	P	R	F1	P	R	F1
Ri-Rc	73.83	69.30	71.49	65.22	66.35	65.75
Re	74.60	69.81	72.10	67.71	67.37	67.52

Table 3: The result of *Rel$^+$-C* for experiments of Ri-Rc and Re with gold entities.

predicted boundaries of its endpoints and the correctness of both entities and relation types (or relation existence).

Implementation details Following previous work, we use *bert-base-uncased* (Devlin et al., 2019) for experiments on ACE2005 and *scibert-scivocab-uncased* (Beltagy et al., 2019) for experiments on SciERC. We consider max span length $L = 8$ for ACE2005 and $L = 12$ for SciERC. We run each experiment setting six times and report the average F1 scores. More hyper-parameters and details are in Appendix B. A significance analysis is done with the permutation test for the results of every two approaches and we reject the null hypothesis when $p < 0.05$.

3.2 Experimental results

The main results of the eight approaches are shown in Table 2. We compare the classification F1 scores and the identification results are shown.

The results of purely pipeline approaches Comparing the results of a1-a4, we can see that a4 is the best pipeline approach for almost all classification evaluations. We can conclude that dividing the NER or RE task into pipelines does not help the entire ERE task. From the results of a2, a3 and a4, we could find out that the dividing of RE (a3 vs. a4) leads to a larger performance drop than the dividing of NER (a2 vs. a4). To exclude the effect of error propagation from NER task, we do extra experiments with gold entities for Ri-Rc and Re. The results are shown in Table 3 and Ri-Rc has a large performance drop with Re. Dividing RE task brings a negative effect to the approaches. We guess because the identification and classification are highly correlated sub-tasks, if they are both difficult, then solving them jointly in one module can promote the performance of both. The entity sub-tasks are not so difficult, especially on ACE2005, so the improvement of a3 or a4 over a1 or a2 on Ent-C is not significant.

The results of approaches with cross-task We compare the results of all the joint approaches: a5 to a8. We observe that a8 is better than the other three on almost all the evaluations except for Ent-C on ACE2005. We first compare EcRi and NerRi to EcRe and NerRe. The Ent-C results of a5 are higher than those of a6 on both datasets and a7 is better than a8 on ACE2005, but for the results of Rel-I and Rel$^+$-I, a8 outperforms a7 and a6 outperforms a5 on most evaluations. We can conclude that EcRe and NerRe are better than EcRi and NerRi. We guess it is the reason that the entity labels of a span pair have a stronger correlation with their relation label than with the existence of their relation. Then from the results of a5 vs. a7 and a6 vs. a8, we wish to investigate the effect of a separate entity identifier. For a5 and a7, we cannot clearly judge which is better, but for a6 and a8, a8 significantly outperforms a6 on most evaluations which shows that the separate Ei hurt the performance of the cross-task module for the error propagated to the downstream modules. Ei has much lower performance on SciERC than on ACE2005, so we guess it brings more performance drop on SciERC than on ACE2005. Comparing the results of a1 vs. a5, a2 vs. a6, and we can see that the EcRi and EcRe modules have lower the evaluation results of Ent-C and Rel-I and the performance gap between a1 and a5, a2 and a6 of Rel$^+$on SciERC is also more than on ACE2005. This gives us the insight that the cross-task module

		ACE2005			SciERC		
		P	R	F1	P	R	F1
Pruner	train	34.47	99.89	51.25	25.13	99.98	40.17
	dev	33.89	99.60	50.34	24.92	99.01	39.82
	test	34.17	99.65	50.89	25.35	99.05	40.37
Ei	train	99.92	99.93	99.92	99.91	99.96	99.94
	dev	93.23	94.16	93.69	78.78	82.48	80.58
	test	93.94	94.87	94.41	78.43	79.93	79.17

Table 4: Comparison of Ei and the pruner on both datasets

may be more sensitive to input error than the single-task module at least for our high-order inference.

The comparison of pipeline and joint approaches Comparing all the pipeline and joint approaches, the common pipeline structure a4 is comparable to all the other approaches except for a8. In particular, a4 outperforms a6, which has a similar structure to some previous joint models (Sun et al., 2019), on almost all evaluations. It shows that pipeline and joint approaches could have comparable performance with the same embedding. But even with the same embedding method, the fully joint approach a8 with the pruner has significantly better performance than a4 on Rel$^+$.

3.3 Analysis

To further investigate the effect of the input error on the joint modules, we conduct extra experiments on the following approaches:

- Ei-NerRe: In this approach, we replace the pruner with a pre-trained Ei module in approach a8. We could treat the Ei module as an entity pruner with lower recall but much higher precision compared to the pruner we used (refer to Table 4). Meanwhile, as the NerRe module could identify the existence of entities, it could fix some input errors compared to EcRe in a6.

- NerRe*: It is the NerRe module with no joint inference. NerRe* only shares encoders across NER and RE sub-tasks and it could be treated as a less complex joint module compared to NerRe.

From the results of Ei-EcRe vs. Ei-NerRe in Table 5, we observe that, as NerRe could reduce the impact of wrongly predicted entities, the latter approach has a slight but not significant advantage over the former on SciERC. When there are only a small amount of input errors, NerRe has a significant advantage over Ei-EcRe and Ei-NerRe

	ACE2005			SciERC		
	Ent-C	*Rel-C*	*Rel$^+$-C*	*Ent-C*	*Rel-C*	*Rel$^+$-C*
Ei-EcRe	87.94	66.46	63.81	66.42	46.31	34.84
Ei-NerRe	88.21	66.97	64.65	65.92	46.71	35.05
NerRe	88.51	67.34	64.66	68.01	47.51	37.42

Table 5: The F1 scores of Ri-EcRe, Ri-NerRe and NerRe.

	ACE2005			SciERC		
	Ent-C	*Rel-C*	*Rel$^+$-C*	*Ent-C*	*Rel-C*	*Rel$^+$-C*
Ei-NerRe	88.21	66.97	64.65	65.92	46.71	35.05
Ei-NerRe*	88.25	66.90	64.15	67.34	47.25	36.01
NerRe	88.51	67.34	64.66	68.01	47.51	37.42
NerRe*	88.63	66.67	64.05	67.64	47.02	36.72

Table 6: The F1 scores of NerRe, NerRe* with different entity pruner. "Ei-" means using a pre-trained Ei module; otherwise, we use the pruner.

on SciERC. Surprisingly, the result is different on ACE2005. NerRe and Ei-NerRe achieve comparable performance. This may come from the different recalls of Ei on the two datasets. According to Table 4, Ei has a much lower recall on the SciERC test dataset than on ACE2005 in comparison to the pruner.

For the same reason, we also see a similar phenomenon in the results of NerRe vs. NerRe* in Table 6. Replacing the pre-trained Ei with the pruner, we could find large performance improvement on the *Rel$^+$-C* metric for both NerRe and NerRe* on SciERC. On the other hand, the pruner does not show any advantage over Ei on ACE2005. Ei-NerRe performs much worse on SciERC than on ACE2005 in comparison to Ei-NerRe*, which also shows that the NerRe module is more sensitive to the input error than the NerRe* module.

4 Conclusion

In this paper, we empirically study several pipeline and joint approaches of the ERE task. We find that pipeline approaches could achieve quite competitive results with some joint approaches, but with span pruning and high-order inference, the full joint model could still outperforms the pipeline approaches. We observe that if the tasks have strong correlations, a properly designed joint approach tends to have higher performance.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61976139).

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. 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 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 551–560, Portland, Oregon, USA. Association for Computational Linguistics.

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

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.

Markus Eberts and Adrian Ulges. 2020. Span-based joint entity and relation extraction with transformer pre-training. In *ECAI 2020*, pages 2006–2013. IOS Press.

Zixia Jia, Zhaohui Yan, and Kewei Tu. 2022a. High-order inference for entity recognition, relation extraction. and event extraction. In *Technical report*.

Zixia Jia, Zhaohui Yan, Haoyi Wu, and Kewei Tu. 2022b. Span-based semantic role labeling with argument pruning and second-order inference. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 10822–10830. AAAI Press.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar. Association for Computational Linguistics.

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 *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1361–1370, Florence, Italy. Association for Computational Linguistics.

Jue Wang and Wei Lu. 2020. Two are better than one: Joint entity and relation extraction with table-sequence encoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1706–1721, Online. Association for Computational Linguistics.

Zhiheng Yan, Chong Zhang, Jinlan Fu, Qi Zhang, and Zhongyu Wei. 2021. A partition filter network for joint entity and relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 185–197, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Deming Ye, Yankai Lin, Peng Li, and Maosong Sun. 2022. Packed levitated marker for entity and relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4904–4917, Dublin, Ireland. Association for Computational Linguistics.

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 *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, Vancouver, Canada. Association for Computational Linguistics.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 50–61, Online. Association for Computational Linguistics.

A Pruner

Pruning strategy For a given token sequence $X = \{x_1, x_2, ..., x_n\}$, the pruner scores the existence for each possible spans with the length limitation L. We rank the spans by their scores and filter out top K as the candidate spans. Basically we filter out the spans according to a ratio to the length of the sentence. As the gold span number is not strict linear with the sentence length, there is a upper bound of the gold span number for each sentence. We set a upper limit m_u for candidate span number of each sentence and a lower limit m_l to avoid the zero candidate span for very short sentences. So the number of candidate spans of a sentence length n is $K = \max(m_l, \min(m_u, \alpha * n))$, where α is the top-K ratio. For both ACE2005 and SciERC datasets, we take $\alpha = 0.5, m_l = 3, m_u = 18$.

Span representation and scoring The model first embed each token, then products the span representations by the tokens inside the spans. The first sub-token embeddings from the last layer of a pre-trained language model is used as the contextualized representation $\mathbf{x}_i$ for each token x_i.

We use two kinds of embedding layers: bi-affine and self-attention pooling for span encoding. For a span s_i with its tokens $(x_{\text{START}}(i), ..., x_{\text{END}}(i))$, its bi-affine representation is a d_{biaf}-dimension vector:

$$\mathbf{h}_b(s_i) = [\mathbf{x}_{\text{START}}(i); 1]^\top W_b [\mathbf{x}_{\text{END}}(i); 1]$$

The self-attention pooling function use the span's token representations as the keys and values, and a linear layer scores the keys to get the weight of the values.

$$w_j \propto \text{Linear}_{att}(\mathbf{x}_j)$$
$$\mathbf{h}_a(s_i) = \sum_{\text{START}(i) \le j \le \text{END}(i)} w_j \mathbf{x}_j$$

Then a two-layers MLP projects the concatenation of these representations into a d_{cat}-dimension hidden space for the final span representation:

$$\mathbf{h}(s_i) = \text{MLP}([\mathbf{h}_b(s_i); \mathbf{h}_a(s_i)])$$

The span representation of s_i is feed into a linear layer to get the score g_i:

$$g_i = \text{Linear}(\mathbf{h}(s_i))$$

Training and evaluation We train the pruner as an identifier, the training loss is the binary cross-entropy:

$$Loss = -\sum_i p_i \log(q_i) + (1 - p_i) \log(1 - q_i)$$

$p_i = 1$ if the span s_i is an entity span otherwise $p_i = 0$ and $q_i = \text{Sigmoid}(g_i)$.

For the evaluation, the pruner produces a candidate span set and calculates the f1 score. We choose the best model on dev sets.

B Hyper-parameters and Implementation Details

We tune the hidden size of MLP_{span} and MLP_{rel} among $[200, 300, 400]$ for each module. The learning rate is tuned among [1e-5, 2e-5, 5e-5] and dropout rate is tuned among [0.1,0.2,0.3].

Setting		Value
Pruner		
	d_{biaf}	768
	d_{cat}	768
	α	0.5
	m_l	3
	m_u	18
Modules encode		
	MLP_{span}	200
	MLP_{rel}	400
High-order inference		
	iterate step	3
	d_t	200
Other settings		
	epochs for `Ei`, `Ec`, `Ri`,`Rc`, `Ner`, `Re`	200(SciERC) 100(ACE2005)
	epochs for `EcRi`, `EcRe`	200(SciERC) 100(ACE2005)
	epochs for `NerRe`	300(SciERC) 200(ACE2005)
	batch size	20
	dropout rate	0.1
	learning rate	1e-5(SciERC) 2e-5(ACE2005)
	lr decay	1e-05
	warm-up rate	0.5
	gradient clipping	5

Table 7: Summary of hyper-parameters