

End-to-end neural relation extraction using deep biaffine attention

Dat Quoc Nguyen and Karin Verspoor

The University of Melbourne, Australia
 {dqnguyen, karin.verspoor}@unimelb.edu.au

Abstract. We propose a neural network model for joint extraction of named entities and relations between them, without any hand-crafted features. The key contribution of our model is to extend a BiLSTM-CRF-based entity recognition model with a deep biaffine attention layer to model second-order interactions between latent features for relation classification, specifically attending to the role of an entity in a directional relationship. On the benchmark “relation and entity recognition” dataset CoNLL04, experimental results show that our model outperforms previous models, producing new state-of-the-art performances.

1 Introduction

Extracting entities and their semantic relations from raw text is a key information extraction task. For example, given the sentence “David Foster is the AP’s Northwest regional reporter, based in Seattle” in the CoNLL04 dataset [27], our goal is to recognize “David Foster” as person, “AP” as organization, and “Northwest” and “Seattle” as location entities, then classify entity pairs to extract structured information: *Work_For*(David Foster, AP), *OrgBased_In*(AP, Northwest) and *OrgBased_In*(AP, Seattle). Such information is useful in many other NLP tasks. Especially in IR applications such as entity search, structured search and question answering, it helps provide end users with significantly better search experience [11,29,6].

A common relation extraction approach is to construct pipeline systems with separate sub-systems for the two tasks of named entity recognition and relation classification [2]. More recently, end-to-end systems which jointly learn to extract entities and relations have been proposed with strong potential to obtain high performance [26]. Traditional joint approaches are feature-based supervised learning methods which employ numerous syntactic and lexical features based on external NLP tools as well as knowledge base resources [12,20,18].

State-of-the-art relation extraction performance has been obtained by end-to-end models based on neural networks. Specifically, Gupta et al. (2016) [9] proposed a RNN-based model which achieved top results on the CoNLL04 dataset. Their approach relies on various manually extracted features. Other neural models employ dependency parsing-based information [19,23,31]. In particular, Miwa and Bansal (2016) [19] applied bottom-up and top-down tree-structured LSTMs to model dependency paths between entities. Zhang et al. (2017) [31] integrated implicit syntactic information by using latent feature representations extracted from a pre-trained BiLSTM-based dependency parser. Zheng et al. (2017) [32] used a softmax layer on top of a BiLSTM for

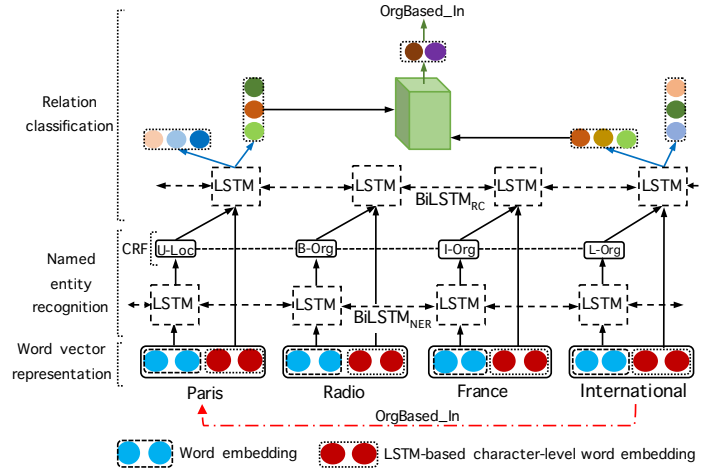


Fig. 1. Illustration of our model. Linear transformations are not shown for simplification.

entity recognition, and a CNN on top of the BiLSTM for classifying relations [22]. Adel and Schütze (2017) [1] assumed that entity boundaries are given, and trained a CNN to extract context features around the entities, and using these features for entity and relation classification. Recently, Wang et al. (2018) [30] formulated the joint entity and relation extraction problem as a directed graph and proposed a BiLSTM- and transition-based approach to generate the graph incrementally. Bekoulis et al. (2018) [4] extended the multi-head selection-based joint model [5] with adversarial training. In [5,33,13], the joint task is formulated as a sequence tagging problem, and a BiLSTM with a softmax output layer can then be used for joint prediction.

In this paper, we present a novel end-to-end neural model for joint entity and relation extraction. As illustrated in Figure 1, our model architecture can be viewed as a mixture of a named entity recognition (NER) component and a relation classification (RC) component. Our NER component employs a BiLSTM-CRF architecture [10] to predict entities from input word tokens. Based on both the input words and the predicted NER labels, the RC component uses another BiLSTM to learn latent features relevant for relation classification. In most previous neural joint models, the relation classification part relies on a common “linear” concatenation-based mechanism over the latent features associated with entity pairs, i.e. the latent features are first concatenated into a single feature vector which is then linearly transformed before being fed into a softmax classifier. In contrast, our RC component takes into account second-order interactions over the latent features via a tensor. In particular, for relation classification we propose a novel use of the deep *biaffine* attention mechanism [7] which was first introduced in dependency parsing.

Experimental results on the benchmark “relation and entity recognition” dataset CoNLL04 [27] show that our model outperforms previous models, obtaining new state-of-the-art scores. In addition, using the biaffine attention improves the performance compared to using the linear mechanism significantly. We also provide an ablation study to investigate effects of different contributing factors in our model.

2 Our proposed model

This section details our end-to-end relation extraction model. Given an input sequence of n word tokens w_1, w_2, \dots, w_n , we use a vector \mathbf{v}_i to represent each i^{th} word w_i by concatenating word embedding $\mathbf{e}_{w_i}^{(w)}$ and character-level word embedding $\mathbf{e}_{w_i}^{(c)}$:

$$\mathbf{v}_i = \mathbf{e}_{w_i}^{(w)} \circ \mathbf{e}_{w_i}^{(c)} \quad (1)$$

Here, for each word type w , we use a one-layer BiLSTM ($\text{BiLSTM}_{\text{char}}$) to learn its character-level word embedding $\mathbf{e}_w^{(c)}$ [3].

Named entity recognition (NER): The NER component feeds the sequence of vectors $\mathbf{v}_{1:n}$ with an additional context position index i into another BiLSTM ($\text{BiLSTM}_{\text{NER}}$) to learn a “latent” feature vector representing the i^{th} word token. Then the NER component performs linear transformation of each latent feature vector by using a single-layer feed-forward network (FFNN_{NER}):

$$\mathbf{h}_i = \text{FFNN}_{\text{NER}}(\text{BiLSTM}_{\text{NER}}(\mathbf{v}_{1:n}, i)) \quad (2)$$

The output layer size of FFNN_{NER} is the number of BIOES-style NER labels [25]. The NER component feeds the output vectors $\mathbf{h}_{1:n}$ into a linear-chain CRF layer [16] for NER label prediction. A cross-entropy loss \mathcal{L}_{NER} is computed during training, while the Viterbi algorithm is used for decoding. Our NER component thus is the BiLSTM-CRF model [10] with additional LSTM-based character-level word embeddings [17].

Relation classification (RC): Assume that t_1, t_2, \dots, t_n are NER labels predicted by the NER component for the input words. We represent each i^{th} predicted label by a vector embedding \mathbf{e}_{t_i} . We create a sequence of vectors $\mathbf{x}_{1:n}$ in which each \mathbf{x}_i is computed as:

$$\mathbf{x}_i = \mathbf{e}_{t_i} \circ \mathbf{v}_i \quad (3)$$

As for NER, the RC component also uses a BiLSTM ($\text{BiLSTM}_{\text{RC}}$) to learn another set of latent feature vectors, but from the sequence $\mathbf{x}_{1:n}$:

$$\mathbf{r}_i = \text{BiLSTM}_{\text{RC}}(\mathbf{x}_{1:n}, i) \quad (4)$$

The RC component further uses these latent vectors \mathbf{r}_i for relation classification.

We propose a novel use of the deep *biaffine* attention mechanism [7] for relation classification. The biaffine attention mechanism was proposed for dependency parsing [7], helping to produce the best reported parsing performance to date [8]. First, to encode the directionality of a relation, we use two single-layer feed-forward networks to project each \mathbf{r}_i into *head* and *tail* vector representations which correspond to whether the i^{th} word serves as the head or tail argument of the relation:

$$\mathbf{h}_i^{(\text{head})} = \text{FFNN}_{\text{head}}(\mathbf{r}_i) \quad (5)$$

$$\mathbf{h}_i^{(\text{tail})} = \text{FFNN}_{\text{tail}}(\mathbf{r}_i) \quad (6)$$

Following [19], our RC component incrementally constructs relation candidates using all possible combinations of the last word tokens of predicted entities, i.e. words

with L or U labels. We assign an entity pair to a negative relation class (NEG) when the pair has no relation or when the predicted entities are not correct. For example, for Figure 1, we would have two relation candidates: *NEG*(Paris, International) and *OrgBased_In*(International, Paris). Then for each head-tail candidate pair (w_j, w_k) , we apply the biaffine attention operator:

$$\mathbf{s}_{j,k} = \text{Biaffine}(\mathbf{h}_j^{(\text{head})}, \mathbf{h}_k^{(\text{tail})}) \quad (7)$$

$$\text{Biaffine}(\mathbf{y}_1, \mathbf{y}_2) = \underbrace{\mathbf{y}_1^T \mathbf{U} \mathbf{y}_2}_{\text{Bilinear}} + \underbrace{\mathbf{W}(\mathbf{y}_1 \circ \mathbf{y}_2) + \mathbf{b}}_{\text{Linear}} \quad (8)$$

where \mathbf{U} , \mathbf{W} , \mathbf{b} are a $m \times l \times m$ tensor, a $l \times (2 \times m)$ matrix and a bias vector, respectively. Here, m is the size of the output layers of both $\text{FFNN}_{\text{head}}$ and $\text{FFNN}_{\text{tail}}$, while l is the number of relation classes (including NEG). Next, the RC component feeds the output vectors $\mathbf{s}_{j,k}$ of the biaffine attention layer into a softmax layer for relation prediction. Another cross-entropy loss \mathcal{L}_{RC} is then computed during training.

Joint learning: The objective loss of our joint model is the sum of the NER and RC losses: $\mathcal{L} = \mathcal{L}_{\text{NER}} + \mathcal{L}_{\text{RC}}$. Model parameters are then learned to minimize \mathcal{L} .

3 Experiments

3.1 Experimental setup

Evaluation scenarios: We evaluate our joint model on two evaluation setup scenarios: **(1) NER&RC:** A realistic scenario where entity boundaries are *not* given. **(2) EC&RC:** A less realistic scenario where the entity boundaries are given [26,12,20]. Thus the NER task which identifies both entity boundaries and classes reduces to the *entity classification* (EC) task. Following [20], we encode the gold entity boundaries in the BIOESU scheme. Then we represent each B, I, O, L or U boundary tag as a vector embedding. As a result, the vector \mathbf{v}_i in Equation 1 now also includes the boundary tag embedding in addition to the word embedding and character-level word embedding.

Dataset: We use the benchmark “entity and relation recognition” dataset CoNLL04 from [27]. Following [4,5], we use the 64%/16%/20% training/development/test pre-split available from Adel and Schütze (2017) [1], in which the test set was previously also used by Gupta et al. (2016) [9].

Implementation: Our model is implemented using DYNET v2.0 [21]. We optimize the objective loss using Adam [14], no mini-batches and run for 100 epochs. We compute the average of NER/EC score and RC score after each training epoch. We choose the model with the highest average score on the development set, which is then applied to the test set for the final evaluation phase. More details of the implementation as well as optimal hyper-parameters are in the Appendix. Our code is available at: <https://github.com/datquocnguyen/jointRE>

Metric: Similar to previous works in Table 1, we use the macro-averaged F1-score over the entity classes to score NER/EC and over the relation classes to score RC. More details of the metric are also in the Appendix. Unlike previous neural models, we report results as mean and standard deviation of the scores over 10 runs with 10 random seeds.

Table 1. Comparison with the previous state-of-the-art results on the **test** set. Recall that Setup 2 uses gold entity boundaries while Setup 1 does not. The subscript denotes the standard deviation. (F) refers to the use of extra feature types such as POS tag-based or dependency parsing-based features. Although using the same test set, Gupta et al. (2016) [9] reported results on a 80/0/20 training/development/test split rather than our 64/16/20 split. Results in the last two rows are just for reference, not for comparison, due to a random sampling of the test set. In particular, Miwa and Sasaki (2014) [20] used the 80/0/20 split for Setup 1 and performed 5-fold cross validation (i.e. sort of equivalent to 80/0/20) for Setup 2, while Zhang et al. (2017) [31] used a 72/8/20 split.

Model	Setup 1		Setup 2	
	NER	RC	EC	RC
Gupta et al. (2016) [9]	-	-	88.8	58.3
Gupta et al. (2016) [9] (F)	-	-	92.4	69.9
Adel and Schütze (2017) [1]	-	-	82.1	62.5
Bekoulis et al. (2018) [4]	83.6	62.0	93.0	68.0
Bekoulis et al. (2018) [5]	83.9	62.0	93.3	67.0
Our joint model	86.2 _{0.5}	64.4 _{0.6}	93.8 _{0.4}	69.6 _{0.7}
Miwa and Sasaki (2014) [20] (F)	80.7	61.0	92.3	71.0
Zhang et al. (2017) [31] (F)	85.6	67.8	-	-

3.2 Main results

End-to-end results: The first six rows in Table 1 compare our results with previous state-of-the-art published results on the same test set. In particular, our model obtains 2+% absolute higher NER and RC scores (Setup 1) than the BiLSTM-CRF-based multi-head selection model [5]. We also obtain 7+% higher EC and RC scores (Setup 2) than Adel and Schütze (2017) [1]. Note that Gupta et al. (2016) [9] use the same test set as we do, however they report final results on a 80/0/20 training/development/test split rather than our 64/16/20, i.e. Gupta et al. (2016) use a larger training set, but producing about 1.5% lower EC score and similar RC score against ours. These results show that our model performs better than previous state-of-the-art models, using the same setup.

In Table 1, the last two rows present results reported in [20] and [31] on the dataset CoNLL04. However, these results are *not* comparable due to their random sampling of the test set, i.e. using different train-test splits. Both Miwa and Sasaki (2014) [20] and Zhang et al. (2017) [31] employ additional extra features based on external NLP tools and use larger training sets than ours. Specifically, Zhang et al. (2017) integrate syntactic features by using a pre-trained BiLSTM-based dependency parser to extract BiLSTM-based latent feature representations for words in the input sentence, and then using these latent representations directly as part of the input embeddings in their model. We plan to extend our model with their syntactic integration approach to further improve our model performance in future work.

Ablation analysis: We provide in Table 2 the results of a pipeline approach where we treat our two NER and RC components as independent networks, and train them separately. Here, the RC network uses gold NER labels when training, and uses predicted labels produced by the NER network when decoding. We find that the joint approach does slightly better than the pipeline approach in relation classification, although the

Table 2. Ablation results on the **development** set. * and ** denote the statistically significant differences against the **full** results at $p < 0.05$ and $p < 0.01$, respectively (using the two-tailed paired t-test). (a) Without using the character-level word embeddings. (b) Using a softmax layer for NER label prediction instead of the CRF layer. (c) Without using the NER label embeddings in our RC component, i.e. Equation 3 would become $\mathbf{x}_i = \mathbf{v}_i$. (d) Without using the Bilinear part in Equation 8, i.e., Biaffine would be a common Linear mechanism. (e) Without using the Linear part in Equation 8, i.e., Biaffine reduces to Bilinear.

Model	Setup 1		Setup 2	
	NER	RC	EC	RC
Pipeline	87.3 _{0.6}	66.3 _{0.8}	93.4 _{0.6}	72.9 _{0.6}
Joint model (full)	87.1 _{0.5}	66.9 _{0.8}	93.3 _{0.5}	73.3 _{0.6}
(a) w/o Character	82.7 _{0.5} **	63.0 _{0.7} **	93.1 _{0.6}	73.4 _{0.8}
(b) w/o CRF	86.4 _{0.5} *	66.0 _{0.8} *	93.5 _{0.4}	73.2 _{0.6}
(c) w/o Entity	87.1 _{0.5}	64.7 _{0.9} *	93.3 _{0.6}	72.1 _{0.7} *
(d) w/o Bilinear	86.6 _{0.5}	65.4 _{0.7} **	93.4 _{0.5}	72.0 _{0.7} **
(e) w/o Linear	86.8 _{0.6}	65.9 _{0.7} *	93.3 _{0.5}	72.0 _{0.5} *

differences are not significant. A similar observation is also found in [19]. Also, in preliminary experiments, we do not find any significant difference in performance of our joint model when feeding gold NER labels instead of predicted NER labels into the RC component during training. This is not surprising as the training NER score is at 99+%.

Table 2 also presents ablation tests over 5 factors of our joint model on the development set. In particular, Setup 1 performances significantly degrade by 4+% absolutely, when not using the character-level word embeddings. The performances also decrease when using a softmax classifier for NER label prediction rather than a CRF layer (here, the decrease is significant). In contrast, we do not find any significant difference in Setup 2 scores when not using either the character-level embeddings or the CRF layer, clearly showing the usefulness of the given gold entity boundaries. The 3 remaining factors, including removing NER label embeddings and not taking either the Bilinear or Linear part (in Equation 8) into the Biaffine attention layer, do not affect the NER/EC score. However, they significantly decrease the RC score. This is reasonable because those 3 factors are part of the RC component only, thus helpful in predicting relations. More specifically, using the Biaffine attention produces about 1.5% significant improvements to a common Linear transformation mechanism in relation classification, i.e., “w/o Bilinear” results against the full results in Table 2: 65.4% vs. 66.9% and 72.0% vs. 73.3% (although using Biaffine increases training time over using Linear by 35%, relatively).

4 Conclusion

In this paper, we have presented an end-to-end neural network-based relation extraction model. Our model employs a BiLSTM-CRF architecture for entity recognition and a biaffine attention mechanism for relation classification. On the benchmark CoNLL04 dataset, our model produces new state-of-the-art performance.

Acknowledgments: This work was supported by the ARC projects DP150101550 and LP160101469.

References

1. Adel, H., Schütze, H.: Global Normalization of Convolutional Neural Networks for Joint Entity and Relation Classification. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pp. 1723–1729 (2017)
2. Bach, N., Badaskar, S.: A Review of Relation Extraction. Tech. rep., Carnegie Mellon University (2007)
3. Ballesteros, M., Dyer, C., Smith, N.A.: Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pp. 349–359 (2015)
4. Bekoulis, G., Deleu, J., Demeester, T., Develder, C.: Adversarial training for multi-context joint entity and relation extraction. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pp. 2830–2836 (2018)
5. Bekoulis, G., Deleu, J., Demeester, T., Develder, C.: Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications* **114**, 34 – 45 (2018)
6. Blanco, R., Cambazoglu, B.B., Mika, P., Torzec, N.: Entity Recommendations in Web Search. In: *Proceedings of the 12th International Semantic Web Conference - Part II*. pp. 33–48 (2013)
7. Dozat, T., Manning, C.D.: Deep Biaffine Attention for Neural Dependency Parsing. In: *Proceedings of the 5th International Conference on Learning Representations* (2017)
8. Dozat, T., Qi, P., Manning, C.D.: Stanford’s Graph-based Neural Dependency Parser at the CoNLL 2017 Shared Task. In: *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. pp. 20–30 (2017)
9. Gupta, P., Schütze, H., Andrassy, B.: Table Filling Multi-Task Recurrent Neural Network for Joint Entity and Relation Extraction. In: *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*. pp. 2537–2547 (2016)
10. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv preprint arXiv:1508.01991* (2015)
11. Jiang, J.: Information Extraction from Text. In: Aggarwal, C.C., Zhai, C. (eds.) *Mining Text Data*, pp. 11–41 (2012)
12. Kate, R.J., Mooney, R.J.: Joint Entity and Relation Extraction Using Card-pyramid Parsing. In: *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. pp. 203–212 (2010)
13. Katiyar, A., Cardie, C.: Going out on a limb: Joint Extraction of Entity Mentions and Relations without Dependency Trees. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 917–928 (2017)
14. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. *CoRR* **abs/1412.6980** (2014)
15. Kiperwasser, E., Goldberg, Y.: Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of ACL* **4**, 313–327 (2016)
16. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. pp. 282–289 (2001)
17. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural Architectures for Named Entity Recognition. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pp. 260–270 (2016)
18. Li, Q., Ji, H.: Incremental Joint Extraction of Entity Mentions and Relations. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 402–412 (2014)

19. Miwa, M., Bansal, M.: 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)*. pp. 1105–1116 (2016)
20. Miwa, M., Sasaki, Y.: Modeling Joint Entity and Relation Extraction with Table Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pp. 1858–1869 (2014)
21. Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., Balles-teros, M., Chiang, D., Clothiaux, D., Cohn, T., Duh, K., Faruqui, M., Gan, C., Garrette, D., Ji, Y., Kong, L., Kuncoro, A., Kumar, G., Malaviya, C., Michel, P., Oda, Y., Richardson, M., Saphra, N., Swayamdipta, S., Yin, P.: DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980* (2017)
22. Nguyen, T.H., Grishman, R.: Combining Neural Networks and Log-linear Models to Improve Relation Extraction. In: *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence* (2016)
23. Pawar, S., Bhattacharyya, P., Palshikar, G.: End-to-end relation extraction using neural networks and markov logic networks. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. pp. 818–827 (2017)
24. Pennington, J., Socher, R., Manning, C.: Glove: Global Vectors for Word Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pp. 1532–1543 (2014)
25. Ratnikov, L., Roth, D.: Design Challenges and Misconceptions in Named Entity Recognition. In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. pp. 147–155 (2009)
26. Roth, D., Yih, W.: Global Inference for Entity and Relation Identification via a Linear Programming Formulation. In: *Introduction to Statistical Relational Learning*. MIT Press (2007)
27. Roth, D., Yih, W.: A Linear Programming Formulation for Global Inference in Natural Language Tasks. In: *Proceedings of the 8th Conference on Computational Natural Language Learning*. pp. 1–8 (2004)
28. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* **15**, 1929–1958 (2014)
29. Thomas, P., Starlinger, J., Vowinkel, A., Arzt, S., Leser, U.: GeneView: A comprehensive semantic search engine for PubMed. *Nucleic Acids Research* **40**(W1), W585–W591 (2012)
30. Wang, S., Zhang, Y., Che, W., Liu, T.: Joint Extraction of Entities and Relations Based on a Novel Graph Scheme. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. pp. 4461–4467 (2018)
31. Zhang, M., Zhang, Y., Fu, G.: End-to-End Neural Relation Extraction with Global Optimization. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pp. 1730–1740 (2017)
32. Zheng, S., Hao, Y., Lu, D., Bao, H., Xu, J., Hao, H., Xu, B.: Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing* **257**, 59–66 (2017)
33. Zheng, S., Wang, F., Bao, H., Hao, Y., Zhou, P., Xu, B.: 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)*. pp. 1227–1236 (2017)

Appendix

Implementation details: We apply dropout [28] with a 67% keep probability to the inputs of BiLSTMs and FFNNs. Following [15], we also use *word dropout* to learn an embedding for unknown words: we replace each word token w appearing $\#(w)$ times in the training set with a special “unk” symbol with probability $p_{unk}(w) = \frac{0.25}{0.25 + \#(w)}$.

Word embeddings are initialized by the 100-dimensional pre-trained GloVe word vectors [24], while character and NER label embeddings are initialized randomly. All these embeddings are then updated during training. For learning character-level word embeddings, we set the size of LSTM hidden states in BiLSTM_{char} to be equal to the size of character embeddings. Here, we perform a minimal grid search of hyper-parameters for Setup 1, resulting in the Adam initial learning rate of 0.0005, the character embedding size of 25, the NER label embedding size of 100, the size of the output layers of both FFNN_{head} and FFNN_{tail} at 100, the number of BiLSTM_{NER} and BiLSTM_{RC} layers at 2 and the size of LSTM hidden states in each layer at 100. These optimal hyper-parameters for Setup 1 are then reused for Setup 2 where we additionally use the boundary tag embedding size of 100.

Metric: Similar to the previous works, when computing the macro-averaged F1 scores, we omit the entity label “Other” and the negative relation “NEG”. Here, for NER an entity is predicted correctly if both the entity boundaries and the entity type are correct, while for EC a multi-token entity is considered as correct if at least one of its comprising tokens is predicted correctly. In all cases, a relation is scored as correct if both the argument entities and the relation type are correct.