An Embarrassingly Easy but Strong Baseline for Nested Named Entity Recognition

Hang Yan, Yu Sun, Xiaonan Li, Xipeng Qiu*
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
{hyan19,lixn20,xpqiu}@fudan.edu.cn
yusun21@m.fudan.edu.cn

Abstract

Named entity recognition (NER) is the task to detect and classify the entity spans in the text. When entity spans overlap between each other, this problem is named as nested NER. Span-based methods have been widely used to tackle the nested NER. Most of these methods will get a score $n \times n$ matrix, where n means the length of sentence, and each entry corresponds to a span. However, previous work ignores spatial relations in the score matrix. In this paper, we propose using Convolutional Neural Network (CNN) to model these spatial relations in the score matrix. Despite being simple, experiments in three commonly used nested NER datasets show that our model surpasses several recently proposed methods with the same pre-trained encoders. Further analysis shows that using CNN can help the model find nested entities more accurately. Besides, we found that different papers used different sentence tokenizations for the three nested NER datasets, which will influence the comparison. Thus, we release a pre-processing script to facilitate future comparison[1].

1 Introduction

Named Entity Recognition (NER) is the task to extract entities from raw text. It has been a fundamental task in the Natural Language Processing (NLP) field. Previously, this task is mainly solved by the sequence labeling paradigm through assigning a label to each token (Huang et al., 2015; Ma and Hovy, 2016; Yan et al., 2019). However, this method is not directly applicable to the nested NER scenario, since a token may be included in two or more entities. To overcome this issue, the span-based method which assigns labels to each span was introduced (Eberts and Ulges, 2020; Li et al., 2020; Yu et al., 2020).

*Corresponding author.

[1]Code is available at https://github.com/yhcc/CNN_Nested_NER.

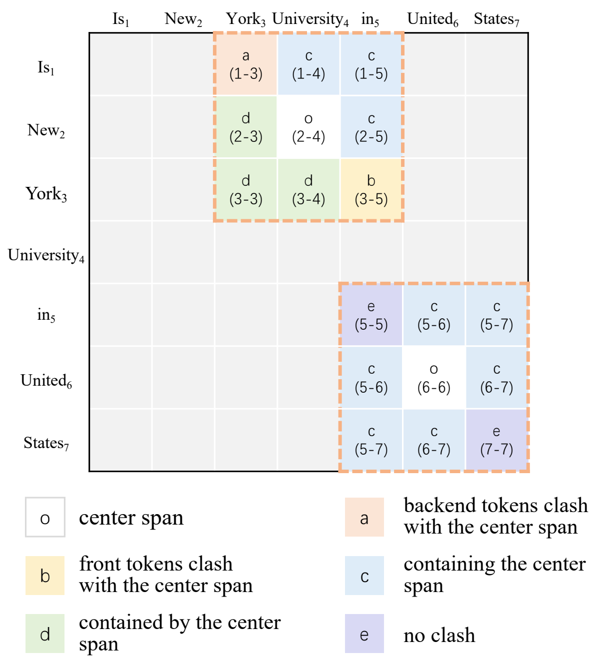

Figure 1: All valid spans of a sentence. We use the start and end tokens to pinpoint a span, for instance, "(2-4)" represents "New York University". Spans in the two orange dotted squares indicates that the center span can have the special relationship (different relations are depicted in different colors) with its surrounding spans. For example, the span "New York" (2-3) is contained by the span "New York University" (2-4). Therefore, the "(2-3)" span is annotated as "d".

Eberts and Ulges (2020) used a pooling method over token representations to get the span representation, and then conducted classification on this span representation. Li et al. (2020) transformed the NER task into a Machine Reading Comprehension form, they used the entity type as the query, and asked the model to select the spans that belong to this entity type. Yu et al. (2020) utilized the Biaffine decoder from dependency parsing (Dozat and Manning, 2017) to convert the span classification into classifying the start and end token pairs. However, these work did not take advantage of the spatial correlations between adjacent spans.

As depicted in Figure 1, the spans surrounding a span have special relationships with the center span. It should be beneficial if we can leverage these spatial correlations. In this paper, we use the Biaffine decoder (Dozat and Manning, 2017) to get a 3D feature matrix, where each entry represents one span. After that, we view this feature matrix as an image and utilize Convolutional Neural Network (CNN) to model the local interaction between spans.

We compare this simple method with recently proposed methods (Wan et al., 2022; Li et al., 2022; Zhu and Li, 2022; Yuan et al., 2022). To make sure our method is strictly comparable to theirs, we asked the authors for their version of data. Although all of them used the same datasets, we found that the statistics, such as the number of sentences and entities, were not the same. This was caused by the usage of distinct sentence tokenization methods, which will influence the performance as shown in our experiments. To facilitate future comparison, we release a pre-processing script[2] for ACE2004, ACE2005 and Genia datasets.

Our contributions can be summarized as follows.

- We find that the adjacent spans have special correlations between each other, and we propose using CNN to model the interaction between them. Despite being very simple, it achieves a considerable performance boost in three widely used nested NER datasets.

- We release a pre-processing script for the three nested NER datasets to facilitate direct and fair comparison.

- The way we view the span feature matrix as an image shall shed some light on future exploration of span-based methods for nested NER task.

2 Related Work

Previously, four kinds of paradigms have been proposed to solve the nested NER task.

The first one is the sequence labeling framework (Straková et al., 2019), since one token can be contained in more than one entities, the Cartesian product of the entity labels are used. However, the Cartesian labels will suffer from the long-tail issue.

The second one is to use the hypergraph to efficiently represent spans (Lu and Roth, 2015; Muis and Lu, 2016; Katiyar and Cardie, 2018; Wang and Lu, 2018). The shortcoming of this method is the complex decoding.

The third one is the sequence-to-sequence (Seq2Seq) framework (Sutskever et al., 2014; Lewis et al., 2020; Raffel et al., 2020) to generate the entity sequence. The entity sequence can be the entity pointer sequence (Yan et al., 2021; Fei et al., 2021) or the entity text sequence (Lu et al., 2022). Nevertheless, the Seq2Seq method suffers from the time-demanding decoding.

The fourth one is to conduct span classification. Eberts and Ulges (2020) proposed to enumerate all possible spans within a sentence, and use a pooling method to get the span representation. While Yu et al. (2020) proposed to use the start and end tokens of a span to pinpoint the span, and use the Biaffine decoder to get the scores for each span. The span-based methods are friendly to parallelism and the decoding is easy. Therefore, this formulation has been widely adopted (Wan et al., 2022; Zhu and Li, 2022; Li et al., 2022; Yuan et al., 2022). However, the relation between neighbor spans was ignored in previous work.

3 Proposed Method

In this section, we first introduce the nested NER task, then describe how to get the feature matrix. After that, we present the CNN module to model

[2] https://github.com/yhcc/CNN_Nested_NER/tree/master/preprocess

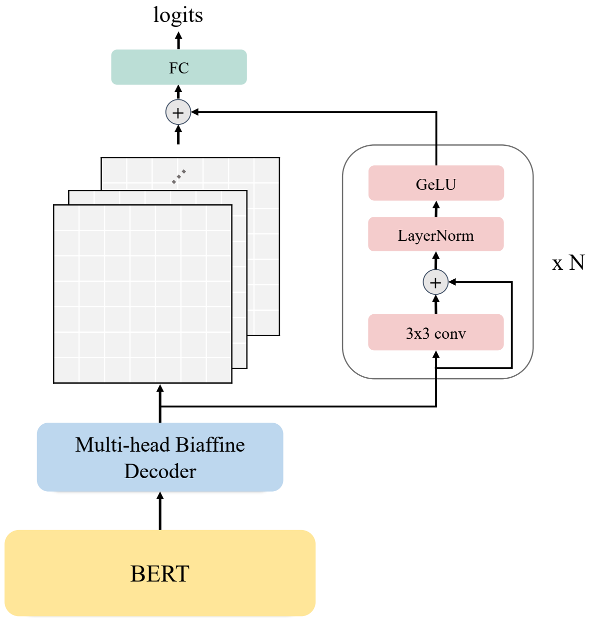

Figure 2: The proposed method in this paper. Use several blocks of CNN to model the spatial correlations between neighbor spans.

the spatial correlation on the feature matrix. A general framework of our proposed method can be viewed in Figure 2.

3.1 Nested NER Task

Given an input sentence $X = [x_1, x_2, \ldots, x_n]$ with n tokens, the nested NER task aims to extract all entities in X. Each entity can be expressed as a tuple (s_i, e_i, t_i). s_i, e_i are the start, end index of the entity. $t_i \in \{1, \ldots, |T|\}$ is its entity type and $|T|$ is the number of entity types. As the task name suggests, the entities may overlap with each other, but different entities are not allowed to have crossing boundaries. For a sentence with n tokens, there are $n(n+1)/2$ valid spans.

3.2 Span-based Method for Nested NER

We follow Yu et al. (2020) to formulate this task into a span classification task. Namely, for each valid span, the model assigns an entity label to it. The method first uses an encoder to encode the input sentence as follows:

$$\mathbf{H} = \mathrm{Encoder}(X),$$

where $\mathbf{H} \in \mathcal{R}^{n\times d}$, and d is the hidden size. Various pre-trained models, such as BERT (Devlin et al., 2019), are usually used as the encoder. For the word tokenized into several pieces, we use max-pooling to aggregate from its pieces' hidden states.

After getting the contextualized embedding of tokens, previous work usually would concatenate it with the static word embedding and the character embedding, and then send this combined embedding into a BiLSTM layer (Yu et al., 2020; Wan et al., 2022; Yuan et al., 2022). To make the model less cluttered, we neither use more embeddings, nor the BiLSTM layer.

Next, we use a multi-head Biaffine decoder (Dozat and Manning, 2017; ?) to get the score matrix as follows:

$$\begin{aligned}
\mathbf{H}_s &= \mathrm{LeakyReLU}(\mathbf{H}W_s), \\
\mathbf{H}_e &= \mathrm{LeakyReLU}(\mathbf{H}W_e), \\
\mathbf{R} &= \mathrm{MHBiaffine}(\mathbf{H}_s, \mathbf{H}_e)
\end{aligned}$$

where $W_s, W_e \in \mathcal{R}^{d\times h}$, h is the hidden size, $\mathrm{MHBiaffine}(\cdot,\cdot)$ is the multi-head Biaffine decoder[3], and $\mathbf{R} \in \mathcal{R}^{n\times n\times r}$, r is the feature size. Each cell (i, j) in the $\mathbf{R}$ can be seen as the feature vector $\mathbf{v} \in \mathcal{R}^r$ for the span. And for the lower triangle of $\mathbf{R}$ (where $i > j$), the span contains words from the j-th to the i-th (Therefore, one span will have two entries if it is off-diagonal).

[3]The detailed description is in the Appendix.

3.3 CNN on Score Matrix

As shown in Figure 1, the cell has relations with cells around. Therefore, we propose using CNN to model these interactions. We repeat the following CNN block several times in our model:

$$\begin{aligned}
\mathbf{R}' &= \mathrm{Conv2d}(\mathbf{R}), \\
\mathbf{R}'' &= \mathrm{GeLU}(\mathrm{LayerNorm}(\mathbf{R}' + \mathbf{R})),
\end{aligned}$$

where Conv2d, LayerNorm and GeLU are the 2D CNN, layer normalization (Ba et al., 2016) and GeLU activation function (Hendrycks and Gimpel, 2016). The layer normalization is conducted in the feature dimension. A noticeable fact here is that since the number of tokens n in sentences varies, their $\mathbf{R}$s are of different shapes. To make sure results are the same when $\mathbf{R}$ is processed in batch, the 2D CNN has no bias term, and all the paddings in $\mathbf{R}$ are filled with 0.

After passing through several CNN blocks, the $\mathbf{R}''$ will be further processed by another 2D CNN module.

3.4 The Output

We use a perceptron to get the prediction logits as follows: [4]

$$P = \mathrm{Sigmoid}(W_o(\mathbf{R} + \mathbf{R}'') + b),$$

where $W_o \in \mathcal{R}^{|T|\times r}$, $b \in \mathcal{R}^{|T|}$, $P \in \mathcal{R}^{n\times n\times |T|}$. And then, we use the binary cross entropy to calculate the loss as

$$\mathcal{L}_{BCE} = -\sum_{0\le i,j<n} y_{ij}\log(P_{ij}),$$

unlike previous works that only use the upper triangle part to get the loss (Yu et al., 2020; Zhu and Li, 2022), we use both upper and lower triangles to calculate the loss. The reason is that in order to conduct batch computation, we cannot solely compute the upper triangle part. Since the lower triangle part has been computed, we also use them for the output. The tag for the score matrix is symmetric, namely, the tag in the (i, j)-th entry is the same as in the (j, i)-th.

[4]We did not use the Softmax because in the very rare case (Such as in the ACE2005 dataset), one span can have more than one entity tag.

	# Param. (Million)	ACE2004			ACE2005		
		P	R	F1	P	R	F1
Data from Li et al. (2022)							
W2NER (Li et al., 2022)[BERT-large]	355.4	87.33	87.71	87.52	85.03	88.62	86.79
Ours[BERT-large]	345.1	87.82_{38}	87.40_{20}	$\mathbf{87.61}_{18}$	86.39_{61}	87.24_{34}	$\mathbf{86.82}_{45}$
w.o. CNN[BERT-large]	343.6	86.54_{48}	87.09_{41}	86.81_{21}	84.88_{26}	86.99_{33}	85.92_{27}
Data from Wan et al. (2022)							
SG (Wan et al., 2022)[BERT-base]	112.3	86.70	85.93	86.31	84.37	85.87	85.11
Ours[BERT-base]	110.5	86.85_{61}	86.45_{36}	$\mathbf{86.65}_{22}$	84.94_{49}	85.40_{27}	$\mathbf{85.16}_{16}$
w.o. CNN[BERT-base]	109.1	85.79_{46}	85.78_{12}	85.78_{22}	82.91_{21}	84.89_{23}	83.89_{16}
Data from Zhu and Li (2022)							
BS (Zhu and Li, 2022)[RoBERTa-base]	125.6	88.43	87.53	87.98	86.25	88.07	87.15
Ours[RoBERTa-base]	125.6	87.77_{27}	88.28_{36}	$\mathbf{88.03}_{14}$	86.58_{78}	87.94_{46}	$\mathbf{87.25}_{48}$
w.o. CNN[RoBERTa-base]	125.2	86.71_{27}	87.40_{42}	87.05_{18}	85.48_{39}	87.54_{59}	86.50_{26}
Data from this work							
W2NER[BERT-large]†	355.4	87.17_{11}	87.70_{19}	87.43_{11}	85.78_{30}	87.81_{24}	86.77_{21}
Ours[BERT-large]	345.1	87.98_{30}	87.50_{22}	$\mathbf{87.74}_{16}$	86.26_{65}	87.56_{31}	$\mathbf{86.91}_{23}$
w.o. CNN[BERT-large]	343.6	86.60_{68}	86.48_{36}	86.54_{19}	84.91_{34}	87.39_{26}	86.13_{30}
BS[RoBERTa-base]†	125.6	87.32_{40}	86.84_{16}	87.08_{24}	86.58_{38}	87.84_{59}	87.20_{32}
Ours[RoBERTa-base]	125.6	87.33_{41}	87.29_{25}	$\mathbf{87.31}_{16}$	86.70_{29}	88.16_{54}	$\mathbf{87.42}_{26}$
w.o. CNN[RoBERTa-base]	125.2	86.09_{36}	86.88_{23}	86.48_{17}	85.17_{67}	88.0_{35}	86.56_{38}

Table 1: Results for the ACE2004 and ACE2005 datasets. Models in the same block of use the same data. The subscript means the standard deviation (e.g 87.73_{18} means 87.73±0.18). † means our reproducation with their publicly available code.

When inference, we calculate scores in the upper triangle part as:

$$\hat{P_{ij}} = (P_{ij} + P_{ji})/2,$$

where $i \leq j$. Then we only use this upper triangle score to get the final prediction. The decoding process generally follows Yu et al. (2020)'s method. We first prune out the non-entity spans (none of its scores is above 0.5), then we sort the remained spans based on their maximum entity score. We pick the spans based on this order, if a span's boundary clashes with selected spans, it is ignored.

4 Experiment

4.1 Experimental Setup

To verify the effectiveness of our proposed method, we conduct experiments in three widely used nested NER datasets, ACE 2004[5] (Doddington et al., 2004), ACE 2005[6] (Walker and Consortium, 2005) and Genia (Kim et al., 2003).

Besides, we choose recently published papers as our baselines. To make sure our experiments are strictly comparable to theirs, we asked the authors for their version of data. The data statistics for each paper are listed in the Appendix. For ACE2004 and ACE2005, although all of them used the same document split as suggested (Lu and Roth, 2015), they used different sentence tokenization, resulting in different numbers of sentences and entities. To facilitate future research on nested NER, we release the pre-processing code and fix some tokenization issues to avoid including unannotated text and dropping entities. While for the Genia data, we fixed some annotation conflicts (the same input sentence with different entity annotation) and dropped several duplicated sentences. The statistics for data processed by us is also presented in the Appendix. We replicate each experiment five times and report its average performance with standard derivation.

4.2 Main Results

Results for ACE2004 and ACE2005 are listed in Table 1, and for Genia is listed in Table 2. When using the same data from previous work, our simple CNN model surpasses the baselines with less or similar parameters, which proves that using CNN to model the interaction between neighbor spans can be beneificial to the nested NER task. Besides, in the bottom block, we reproduced some baselines in our newly processed data to facilitate future comparison. Comparing the last block (processed by us) and the upper blocks (data from previous work), different tokenizations can indeed influence the

[5]https://catalog.ldc.upenn.edu/LDC2005T09

[6]https://catalog.ldc.upenn.edu/LDC2006T06

	# Param. (Million)	Genia P	R	F1
Data from Li et al. (2022)				
W2NER	113.6	83.10	79.76	81.39
Ours	112.6	83.18_{24}	79.70_{8}	$\mathbf{81.40}_{11}$
w.o. CNN	111.1	80.66_{4}	79.76_{7}	80.21_{5}
Data from Wan et al. (2022)				
SG	112.7	77.92	80.74	79.30
Ours	112.2	81.05_{48}	77.87_{65}	$\mathbf{79.42}_{20}$
w.o. CNN	111.1	78.60_{41}	78.35_{52}	78.47_{16}
Data from Yuan et al. (2022)				
Triaffine	526.5	80.42	82.06	81.23
Ours	128.42	83.37_{9}	79.43_{15}	$\mathbf{81.35}_{8}$
w.o. CNN	111.1	80.87_{23}	79.47_{23}	80.16_{16}
Data from this work				
W2NER†	113.6	83.06_{20}	79.42_{20}	81.20_{12}
Ours	112.6	83.05_{15}	79.47_{25}	$\mathbf{81.22}_{14}$
w.o. CNN	111.1	80.63_{17}	79.37_{10}	80.15_{12}

Table 2: Experiment results for the Genia Dataset. "W2NER", "SG" and "Triaffine" are from (Li et al., 2022), (Wan et al., 2022) and (Yuan et al., 2022), all model use the BioBERT-base(Lee et al., 2020). The subscript means the standard deviation (e.g 81.40_{11} means 81.40±0.11). † means our reproduction with their publicly available code.

	FEP	FER	NEP	NER
ACE2004				
Ours	$86.9_{0.2}$	$87.3_{0.5}$	$\mathbf{88.8}_{0.9}$	$88.4_{0.6}$
w.o. CNN	$86.3_{0.8}$	$86.8_{0.3}$	$86.6_{1.3}$	$89.4_{0.8}$
ACE2005				
Ours	$86.2_{0.6}$	$88.3_{0.1}$	$\mathbf{89.0}_{0.8}$	$91.4_{0.5}$
w.o. CNN	$85.2_{0.7}$	$87.9_{0.3}$	$86.2_{0.8}$	$91.3_{0.5}$
Genia				
Ours	$83.4_{0.1}$	$79.8_{0.1}$	$\mathbf{73.3}_{0.6}$	$72.8_{1.8}$
w.o. CNN	$81.1_{0.1}$	$79.4_{0.3}$	$64.1_{0.5}$	$72.1_{0.8}$

Table 3: The precision and recall for flat and nested entities in the test set of three datasets. FEP, FER, NEP and NER are the flat entity precision, flat entity recall, nested entity precision and nested entity recall, respectively. Compared with model without CNN ("w.o. CNN"), the most improved metric is bold. By using CNN, the precision for nested entities improve significantly. The subscript means the standard deviation (e.g $88.8_{0.9}$ means 88.8±0.9).

performance. Therefore, we appeal for the same tokenization for future comparison.

4.3 Why CNN Helps

To study why CNN can boost the performance of the nested NER dataset, we split entities into two kinds. One kind is entities that overlap with other entities, and the other kind is entities that do not. The results of FEP, FER, NEP, and NER[7] are listed in Table 3. Compared with models without CNN module, the NEP of models with CNN module improved for 2.2, 2.8 and 9.2 for ACE2004, ACE2005 and Genia respectively. Namely, much of the performance improvement can be ascribed to finding nested entities more accurately. This is expected as the CNN can be more effective for exploiting the neighbor entities when they are nested.

5 Conclusion

In this paper, we propose using CNN on the score matrix of span-based NER model. Although this method is very simple, it achieves comparable or better performance than recently proposed methods. Analysis shows exploiting the spatial correlation between neighbor spans through CNN can help model find nested entities more precisely. And experiments show that different tokenizations indeed influence the performance. Therefore, it is necessary to make sure all comparative baselines uses the same tokenization. To facilitate future comparison, we release a new pre-processing script for three nested NER datasets.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie M. Strassel, and Ralph M. Weischedel. 2004. The automatic content extraction (ACE) program - tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*. European Language Resources Association.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April*

[7]The detailed description of the four metrics locate in the Appendix.

24-26, 2017, Conference Track Proceedings. OpenReview.net.

Markus Eberts and Adrian Ulges. 2020. Span-based joint entity and relation extraction with transformer pre-training. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2006–2013. IOS Press.

Hao Fei, Donghong Ji, Bobo Li, Yijiang Liu, Yafeng Ren, and Fei Li. 2021. Rethinking boundaries: End-to-end recognition of discontinuous mentions with pointer networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12785–12793. AAAI Press.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 861–871. Association for Computational Linguistics.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. In *Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology, June 29 - July 3, 2003, Brisbane, Australia*, pages 180–182.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinform.*, 36(4):1234–1240.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Jingye Li, Hao Fei, Jiang Liu, Shengqiong Wu, Meishan Zhang, Chong Teng, Donghong Ji, and Fei Li. 2022. Unified named entity recognition as word-word relation classification. 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 10965–10973. AAAI Press.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5849–5859. Association for Computational Linguistics.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 857–867. The Association for Computational Linguistics.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 5755–5772. Association for Computational Linguistics.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Aldrian Obaja Muis and Wei Lu. 2016. Learning to recognize discontiguous entities. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 75–84. The Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*

2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pages 5326–5331. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

C. Walker and Linguistic Data Consortium. 2005. *ACE 2005 Multilingual Training Corpus*. LDC corpora. Linguistic Data Consortium.

Juncheng Wan, Dongyu Ru, Weinan Zhang, and Yong Yu. 2022. Nested named entity recognition with span-level graphs. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 892–903. Association for Computational Linguistics.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 204–214. Association for Computational Linguistics.

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. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. 2019. TENER: adapting transformer encoder for named entity recognition. *CoRR*, abs/1911.04474.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5808–5822. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6470–6476. Association for Computational Linguistics.

Zheng Yuan, Chuanqi Tan, Songfang Huang, and Fei Huang. 2022. Fusing heterogeneous factors with triaffine mechanism for nested named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3174–3186. Association for Computational Linguistics.

Enwei Zhu and Jinpeng Li. 2022. Boundary smoothing for named entity recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 7096–7108. Association for Computational Linguistics.

A Multi-head Biaffine Decoder

The input of Multi-head Biaffine decoder is two matrix $\mathbf{H}_s, \mathbf{H}_e \in \mathcal{R}^{n\times h}$, and the output is $\mathbf{R} \in \mathcal{R}^{n\times n\times r}$. The formulation of Multi-head Biaffine decoder is as follows

$$
\begin{aligned}
\mathbf{S}_1[i,j] &= (\mathbf{H}_s[i] \oplus \mathbf{H}_e[j] \oplus \mathbf{w}_{i-j})W, \\
\{\mathbf{H}_s^{(k)}\}, \{\mathbf{H}_e^{(k)}\} &= \mathrm{Split}(\mathbf{H}_s), \mathrm{Split}(\mathbf{H}_e), \\
\mathbf{S}_2^{(k)}[i,j] &= \mathbf{H}_s^{(k)}[i] U \mathbf{H}_e^{(k)}[j]^T, \\
\mathbf{S}_2 &= \mathrm{Concat}(\mathbf{S}_2^{(1)}, ..., \mathbf{S}_2^{(K)}), \\
\mathbf{R} &= \mathbf{S}_1 + \mathbf{S}_2,
\end{aligned}
$$

where $\mathbf{H}_s, \mathbf{H}_e \in \mathcal{R}^{n\times h}$, h is the hidden size, $\mathbf{w}_{i-j} \in \mathcal{R}^c$ is the span length embedding for length $i-j$, $W \in \mathcal{R}^{(2h+c)\times r}$, $\mathbf{S}_1 \in \mathcal{R}^{n\times n\times r}$, r is the biaffine feature size, $\mathrm{Split}(\cdot)$ equally splits a matrix in the last dimension, thus, $\mathbf{H}_s^{(k)}, \mathbf{H}_e^{(k)} \in \mathcal{R}^{n\times h_k}$; h_k is the hidden size for each head, and $U \in \mathcal{R}^{h_k\times r_k\times h_k}$, $\mathbf{S}_2 \in \mathcal{R}^{n\times n\times r}$, and $\mathbf{R} \in \mathcal{R}^{n\times n\times r}$.

We did not use multi-head for W, because it does not occupy too much parameters and using multi-head for W harms the performace slightly.

B Data

We list the statistics for each datasets in Table 4. As shown in the table, the number of sentences and even the number of entities are different. Therefore, it is not fair to directly compare results from different papers. For the ACE2004 and ACE2005, we release the pre-processing code to get data from the LDC files. We make sure no entities are dropped because of sentence tokenization. Thus, the pre-processed ACE2004 and ACE2005 data from this

[8]The number of entites is different from that reported in their paper, because we found some duplicated entities in their data.

		Sentence				Mention				
		#Train	#Dev	#Test	Avg. Len	#Ovlp.	#Train	#Dev	#Test	Avg. Len
ACE2004	W2NER	6,802	813	897	20.12	12,571	22,056	2,492	3,020	2.5
	SG	6.198	742	809	21.55	12,666	22,195	2,514	3,034	2.51
	BS	6,799	829	879	20.43	12,679	22,207	2,511	3,031	2.51
	Ours	6,297	742	824	23.52	12,690	22,231	2,514	3,036	2.64
ACE2005	W2NER	7,606	1,002	1,089	17.77	12,179	24,366	3,188	2,989	2.26
	SG	7,285	968	1,058	18.60	12,316	24,700	3,218	3,029	2.26
	BS	7,336	958	1,047	18.90	12,313	24,687	3,217	3,027	2.26
	Ours	7,178	960	1,051	20.59	12,405	25,300	3,321	3,099	2.40
Genia	W2NER	15,023	1,669	1,854	25.41	10,263	45,144	5,365	5,506	1.97
	SG	15,022	1,669	1,855	26.47	10,412	47,006	4,461	5,596	2.07
	Triaffine	16,692	-	1,854	25.41	10,263	50,509	-	5,506	1.97
	Ours	14,957	1,667	1,850	25.48	10,261	45,133	5,365	5,506	1.97

Table 4: The statistics used in each paper. "W2NER"[8], "SG", "BS" and "Triaffine" are from (Li et al., 2022), (Wan et al., 2022), (Zhu and Li, 2022) and (Yuan et al., 2022), respectively. Different papers used different sentence tokenization for ACE2004 and ACE2005, resulting in different numbers of sentences in each split. To facilitate future comparison, we open-sourced a pre-processing script to prepare ACE2004 and ACE2005. Previously, some entities will be dropped because of sentence tokenization, we avoid sentence tokenization within an entity and resulting in more entities. And for Genia, different papers used different train/dev/test splits. Besides, the Genia data have duplicated data, we remove these repeated sentences. The data annotated with "Ours" is obtained by our pre-processing code.

work in Table 4 have the most entities. And for Genia, we appeal for the usage of train/dev/test, and we release the data split within the code repo. Moreover, after scrutinizing the Genia data, we found that there exist several duplicated and unwanted data, such as *"(ABSTRACT TRUNCATED AT 250 WORDS)"*, and conflict annotation. For duplicated data in each data split, we only keep one. And we also find some conflict annotation in Genia, we manually fix the discrepancy.

C Implementation Details

We used the AadmW optimizer to optimize the model and the transformers package for the pre-trained model (Wolf et al., 2020). The hyper-parameter range in this paper is listed in Table 5.

	ACE2004	ACE2005	Genia
# Epoch	50	50	5
Learning Rate	2e-5	2e-5	7e-6
Batch size	48	48	8
# CNN Blocks	[2, 3]	[2, 3]	3
CNN kernel size	3	3	3
CNN Channel dim.	[120, 200]	[120, 200]	200
# Head	[1, 5]	[1, 5]	4
Hidden size h	200	200	400
Warmup factor	0.1	0.1	0.1

Table 5: The hyper-parameter in this paper.

	# Ent.	# Flat Ent.	# Nested Ent.
ACE2004	3,036	1,614	1,422
ACE2005	3,099	1,913	1,186
Genia	5,506	4,307	1,199

Table 6: The flat and nested entity statistics in the test set of each dataset.

D FEP FER NEP NER

We split entities into two kinds based on whether they overlap with other entities, and the statistics for each dataset are listed in Table 6. When calculating the flat entity precision (FEP), we first get all flat entities in the prediction and calculate their ratio in the gold. For the flat entity recall (FER), we get all flat entities in the gold and calculate their ratio in the prediction. And we get the nested entity precision (NEP) and nested entity recall (NER) similarly.