Deep Span Representations for Named Entity Recognition

Anonymous ACL submission

Abstract

Span-based models are one of the most straightforward methods for named entity recognition (NER). Existing span-based NER systems shallowly aggregate the token representations to span representations. However, this typically results in significant ineffectiveness for long entities, a coupling between the representations of overlapping spans, and ultimately a performance degradation. In this study, we propose DSpERT (**D**eep **Sp**an **E**ncoder **R**epresentations from **T**ransformers), which comprises a standard Transformer and a span Transformer. The latter uses low-layered span representations as queries, and aggregates the token representations as keys and values, layer by layer from bottom to top. Thus, DSpERT produces span representations of deep semantics.

With weight initialization from pretrained language models, DSpERT achieves performance higher than or competitive with recent state-of-the-art systems on six NER benchmarks.[1] Experimental results verify the importance of the depth for span representations, and show that DSpERT performs particularly well on long-span entities and nested structures. Further, the deep span representations are well structured and easily separable in the feature space.

1 Introduction

As a fundamental information extraction task, named entity recognition (NER) requires predicting a set of entities from a piece of text. Thus, the model has to distinguish the entity spans (i.e., positive examples) from the non-entity spans (i.e., negative examples). In this view, it is natural to enumerate all possible spans and classify them into the entity categories (including an extra non-entity category). This is exactly the core idea of span-based approaches (Sohrab and Miwa, 2018; Eberts and Ulges, 2020; Yu et al., 2020).

[1] Our code will be publicly released.

Analogously to how representation learning matters to image classification (Katiyar and Cardie, 2018; Bengio et al., 2013; Chen et al., 2020), it should be crucial to construct good span representations for span-based NER. However, existing models typically build span representations by shallowly aggregating the top/last token representations, e.g., pooling over the sequence dimension (Sohrab and Miwa, 2018; Eberts and Ulges, 2020; Shen et al., 2021), or integrating the starting and ending tokens (Yu et al., 2020; Li et al., 2020c). In that case, the token representations have not been fully interacted before they are fed into the classifier, which impairs the capability of capturing the information of long spans. If the spans overlap, the resulting span representations are technically coupled because of the shared tokens. This causes the representations less distinguishable from the ones of overlapping spans in nested structures.

Inspired by (probably) the most sophisticated implementation of attention mechanism — Transformer and BERT (Vaswani et al., 2017; Devlin et al., 2019), we propose DSpERT, which stands for **D**eep **Sp**an **E**ncoder **R**epresentations from **T**ransformers. It consists of a standard Transformer and a *span Transformer*; the latter uses low-layered span representations as queries, and token representations within the corresponding span as keys and values, and thus aggregates token representations layer by layer from bottom to top. Such multi-layered Transformer-style aggregation promisingly produces *deep span representations* of rich semantics, analogously to how BERT yields highly contextualized token representations.

With weight initialization from pretrained language models (PLMs), DSpERT performs comparably to recent state-of-the-art (SOTA) NER systems on six well-known benchmarks. Experimental results clearly verify the importance of the depth for the span representations. In addition, DSpERT achieves particularly amplified performance im-

provements against its shallow counterparts[2] on long-span entities and nested structures.

Different from most related work which focuses on the decoder designs (Yu et al., 2020; Li et al., 2020b; Shen et al., 2021; Li et al., 2022), we make an effort to optimize the span representations, but employ a simple and standard neural classifier for decoding. This exposes the *pre-logit representations* that directly determine the entity prediction results, and thus allows further representation analysis widely employed in a broader machine learning community (Van der Maaten and Hinton, 2008; Krizhevsky et al., 2012). This sheds light on neural NER systems towards higher robustness and interpretability (Ouchi et al., 2020).

2 Related Work

The NER research had been long-term focused on recognizing flat entities. After the introduction of linear-chain conditional random field (Collobert et al., 2011), neural sequence tagging models became the *de facto* standard solution for flat NER tasks (Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016; Zhang and Yang, 2018).

Recent studies pay much more attention to nested NER, which a plain sequence tagging model struggles with (Ju et al., 2018). This stimulates a number of novel NER system designs beyond the sequence tagging framework. Hypergraph-based methods extend sequence tagging by allowing multiple tags for each token and multiple tag transitions between adjacent tokens, which is compatible with nested structures (Lu and Roth, 2015; Katiyar and Cardie, 2018). Span-based models enumerate candidate spans and classify them into entity categories (Sohrab and Miwa, 2018; Eberts and Ulges, 2020; Yu et al., 2020). Li et al. (2020b) reformulates nested NER as a reading comprehension task. Shen et al. (2021, 2022) borrow the methods from image object detection to solve nested NER. Yan et al. (2021) propose a generative approach, which encodes the ground-truth entity set as a sequence, and thus reformulates NER as a sequence-to-sequence task. Li et al. (2022) describe the entity set by word-word relation, and solve nested NER by word-word relation classification.

[2]In this paper, unless otherwise specified, we use "shallow" to refer to models that construct span representations by shallowly aggregating (typically top) token representations, although the token representations could be "deep".

The span-based models are probably the most straightforward among these approaches. However, existing span-based models typically build span representations by shallowly aggregating the top token representations from a standard text encoder. Here, the shallow aggregation could be pooling over the sequence dimension (Eberts and Ulges, 2020; Shen et al., 2021), integrating the starting and ending token representations (Yu et al., 2020; Li et al., 2020c), or a concatenation of these results (Sohrab and Miwa, 2018). Apparently, shallow aggregation may be too simple to capture the information embedded in long spans; and if the spans overlap, the resulting span representations are technically coupled because of the shared tokens. These ultimately lead to a performance degradation.

Our DSpERT addresses this issue by multi-layered and bottom-to-top construction of span representations. Empirical results show that such deep span representations outperform the shallow counterpart qualitatively and quantitatively.

3 Methods

Deep Token Representations. Given a T-length sequence passed into an L-layered d-dimensional Transformer encoder (Vaswani et al., 2017), the initial token embeddings, together with the potential positional and segmentation embeddings (e.g., BERT; Devlin et al., 2019), are denoted as $\mathbf{H}^0 \in \mathbb{R}^{T\times d}$. Thus, the l-th ($l = 1, 2, \ldots, L$) token representations are:

$$\mathbf{H}^l = \mathrm{TrBlock}(\mathbf{H}^{l-1}, \mathbf{H}^{l-1}, \mathbf{H}^{l-1}), \qquad (1)$$

where $\mathrm{TrBlock}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ is a Transformer encoder block that takes $\mathbf{Q} \in \mathbb{R}^{T\times d}$, $\mathbf{K} \in \mathbb{R}^{T\times d}$, $\mathbf{V} \in \mathbb{R}^{T\times d}$ as the query, key, value inputs, respectively. It consists of a multi-head attention module and a position-wise feed-forward network (FFN), both followed by a residual connection and a layer normalization. Passing a same matrix, i.e., $\mathbf{H}^{l-1}$, for queries, keys and values exactly results in self-attention (Vaswani et al., 2017).

The resulting top representations $\mathbf{H}^L$, computed through L Transformer blocks, are believed to embrace deep, rich and contextualized semantics that are useful for a wide range of tasks. Hence, in a typical neural NLP modeling paradigm, only the top representations $\mathbf{H}^L$ are used for loss calculation and decoding (Devlin et al., 2019; Eberts and Ulges, 2020; Yu et al., 2020).

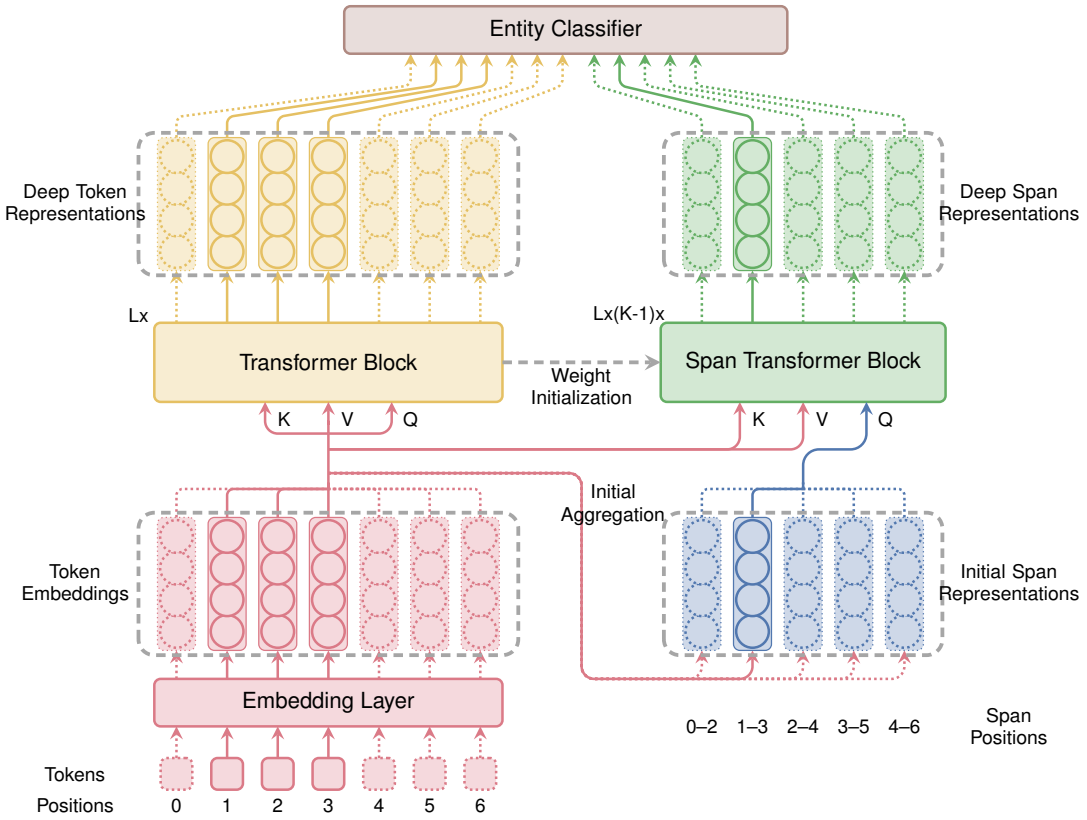


Figure 1: Architecture of DSpERT. It comprises: (*Left*) a standard L-layer Transformer encoder (e.g., BERT); and (*Right*) a span Transformer encoder, where the span representations are the query inputs, and token representations (from the Transformer encoder) are the key/value inputs. There are totally $K-1$ span Transformer encoders, where K is the maximum span size; and each has L layers. The figure specifically displays the case of span size 3; the span of positions 1–3 is highlighted, whereas the others are in dotted lines.

Deep Span Representations. Figure 1 presents the architecture of DSpERT, which consists of a standard Transformer encoder and a *span Transformer encoder*. In a span Transformer of size k ($k = 2, 3, \ldots, K$), the initial span representations $\mathbf{S}^{0,k} \in \mathbb{R}^{(T+k-1)\times d}$ are directly aggregated from the corresponding token embeddings:

$$\boldsymbol{s}_i^{0,k} = \text{Aggregating}(\mathbf{H}^0_{[i:i+k]}), \qquad (2)$$

where $\boldsymbol{s}_i^{0,k} \in \mathbb{R}^d$ is the i-th vector of $\mathbf{S}^{0,k}$, and $\mathbf{H}^0_{[i:i+k]} = [\boldsymbol{h}_i^0; \ldots; \boldsymbol{h}_{i+k-1}^0] \in \mathbb{R}^{k\times d}$ is a slice of $\mathbf{H}^0$ from position i to position $i+k-1$; $\text{Aggregating}(\cdot)$ is a shallowly aggregating function, such as max-pooling. Check Appendix A for more details on alternative aggregating functions used in this study. Technically, $\boldsymbol{s}_i^{0,k}$ covers the token embeddings in the span $(i, i+k)$.

The computation of high-layered span representations imitates that of the standard Transformer. For each span Transformer block, the query is a low-layered span representation vector, and the keys and values are the aforementioned token representation vectors in the positions of that very span. Formally, the l-th layer span representations are:

$$\boldsymbol{s}_i^{l,k} = \text{SpanTrBlock}(\boldsymbol{s}_i^{l-1,k}, \mathbf{H}^{l-1}_{[i:i+k]}, \mathbf{H}^{l-1}_{[i:i+k]}), \qquad (3)$$

where $\text{SpanTrBlock}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ shares the exactly same structure with the corresponding Transformer block, but receives different inputs.[3] More specifically, for span $(i, i+k)$, the query is the span representation $\boldsymbol{s}_i^{l-1,k}$, and the keys and values are the token representations $\mathbf{H}^{l-1}_{[i:i+k]}$. Again, the resulting $\boldsymbol{s}_i^{l,k}$ technically covers the token representations in the span $(i, i+k)$ on layer $l-1$.

The top span representations $\mathbf{S}^{L,k}$ are built through L Transformer blocks, which are capable of enriching the representations towards deep semantics. Thus, the representations of overlapping spans are decoupled, and promisingly distinguishable from each other, although they are originally

[3] In the default configuration, the weights in the span Transformer are independent from those in the Transformer.

built from $\mathbf{S}^{0,k}$ — those shallowly aggregated from token embeddings. This is conceptually analogous to how the BERT uses 12 or more Transformer blocks to produce highly contextualized representations from the original static token embeddings.

The top span representations are then passed to an entity classifier. Note that we do not construct a unigram span Transformer, but directly borrow the token representations as the span representations of size 1. In other words,

$$\mathbf{S}^{L,1} \equiv \mathbf{H}^{L}. \tag{4}$$

Entity Classifier. Following Dozat and Manning (2017) and Yu et al. (2020), we introduce a dimension-reducing FFN before feeding the span representations into the decoder. According to the preceding notations, the representation of span (i, j) is $\boldsymbol{s}_i^{L,j-i}$, thus,

$$\boldsymbol{z}_{ij} = \mathrm{FFN}(\boldsymbol{s}_i^{L,j-i} \oplus \boldsymbol{w}_{j-i}), \tag{5}$$

where $\boldsymbol{w}_{j-i} \in \mathbb{R}^{d_w}$ is the $(j-i)$-th width embedding from a dedicated learnable matrix; $\oplus$ means the concatenation operation. $\boldsymbol{z}_{ij} \in \mathbb{R}^{d_z}$ is the dimension-reduced span representation, which is then fed into a softmax layer:

$$\hat{\boldsymbol{y}}_{ij} = \mathrm{softmax}(\mathbf{W}\boldsymbol{z}_{ij} + \boldsymbol{b}), \tag{6}$$

where $\mathbf{W} \in \mathbb{R}^{c \times d_z}$ and $\boldsymbol{b} \in \mathbb{R}^{c}$ are learnable parameters, and $\hat{\boldsymbol{y}}_{ij} \in \mathbb{R}^{c}$ is the vector of predicted probabilities over entity types. Note that Eq. (6) follows the form of a typical neural classification head, which receives a single vector $\boldsymbol{z}_{ij}$, and yields the predicted probabilities $\hat{\boldsymbol{y}}_{ij}$. Here, the pre-softmax vector $\mathbf{W}\boldsymbol{z}_{ij}$ is called *logits*, and $\boldsymbol{z}_{ij}$ is called *pre-logit representation* (Müller et al., 2019).

Given the one-hot encoded ground truth $\boldsymbol{y}_{ij} \in \mathbb{R}^{c}$, the model can be trained by optimizing the cross entropy loss for all spans:

$$\mathcal{L} = -\sum_{0 \le i < j \le T} \boldsymbol{y}_{ij}^{\mathsf{T}} \log(\hat{\boldsymbol{y}}_{ij}). \tag{7}$$

4 Experiments

4.1 Experimental Settings

Datasets. We perform experiments on four English nested NER datasets: ACE 2004[4], ACE 2005[5], GENIA (Kim et al., 2003) and KBP 2017 (Ji et al., 2017); and two English flat NER datasets: CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) and OntoNotes 5[6]. More details on data processing and descriptive statistics are reported in Appendix B.

Implementation Details. To save space, our implementation details are all placed in Appendix C.

4.2 Main Results

Table 1 shows the evaluation results on English nested NER benchmarks. For a fair and reliable comparison to previous SOTA NER systems, we run DSpERT for five times on each dataset, and report both the best score and the average score with corresponding standard deviation.

With a base-sized PLM, DSpERT achieves on-par or better results compared with previous SOTA systems. More specifically, the best F_1 scores are 88.31%, 87.42%, 81.90% and 87.65% on ACE 2004, ACE 2005, GENIA and KBP 2017, respectively. Except for ACE 2005, these scores correspond to 0.17%, 0.13% and 3.15% absolute improvements.

Table 2 presents the results on English flat NER datasets. The best F_1 scores are 93.70% and 91.76% on CoNLL 2003 and OntoNotes 5, respectively. These scores are slightly higher than those reported by previous literature.

Appendix D further lists the category-wise F_1 scores; the results show that DSpERT can consistently outperform the biaffine model, a classic and strong baseline, across most entity categories. Appendix E provides additional experimental results on Chinese NER, suggesting that the effectiveness of DSpERT is generalizable across languages.

Overall, DSpERT shows strong and competitive performance on both the nested and flat NER tasks. Given the long-term extensive investigation and experiments on these datasets by the NLP community, the seemingly marginal performance improvements are still notable.

4.3 Ablation Studies

We perform ablation studies on three datasets, i.e., ACE 2004, GENIA and CoNLL 2003, covering flat and nested, common and domain-specific corpora.

Depth of Span Representations. As previously highlighted, our core argument is that the deep span representations, which are computed throughout the span Transformer blocks, embrace deep and rich semantics and thus outperform the shallow counterparts.

To validate this point, Table 3 compares DSpERT to the models with a shallow setting,

[4] https://catalog.ldc.upenn.edu/LDC2005T09.
[5] https://catalog.ldc.upenn.edu/LDC2006T06.
[6] https://catalog.ldc.upenn.edu/LDC2013T19.

ACE 2004			
Model	Prec.	Rec.	F1
Li et al. (2020b)	85.05	86.32	85.98
Yu et al. (2020)	87.3	86.0	86.7
Yan et al. (2021)	87.27	86.41	86.84
Shen et al. (2021)	87.44	87.38	87.41
Li et al. (2022)‡	87.33	87.71	87.52
Zhu and Li (2022)	88.43	87.53	87.98
Shen et al. (2022)	88.48	87.81	88.14
DSpERT†	88.29	88.32	**88.31**
DSpERT‡	87.90	88.21	$88.05_{\pm 0.18}$

ACE 2005			
Model	Prec.	Rec.	F1
Li et al. (2020b)	87.16	86.59	86.88
Yu et al. (2020)	85.2	85.6	85.4
Yan et al. (2021)	83.16	86.38	84.74
Shen et al. (2021)	86.09	87.27	86.67
Li et al. (2022)‡	85.03	88.62	86.79
Zhu and Li (2022)	86.25	88.07	87.15
Shen et al. (2022)	86.27	88.60	**87.42**
DSpERT†	87.01	87.84	**87.42**
DSpERT‡	85.73	88.19	$86.93_{\pm 0.49}$

GENIA			
Model	Prec.	Rec.	F1
Yu et al. (2020)*	81.8	79.3	80.5
Yan et al. (2021)	78.87	79.6	79.23
Shen et al. (2021)*	80.19	80.89	80.54
Li et al. (2022)‡	83.10	79.76	81.39
Shen et al. (2022)*	83.24	80.35	81.77
DSpERT†	82.31	81.49	**81.90**
DSpERT‡	81.72	81.21	$81.46_{\pm 0.25}$

KBP 2017			
Model	Prec.	Rec.	F1
Li et al. (2020b)	82.33	77.61	80.97
Shen et al. (2021)	85.46	82.67	84.05
Shen et al. (2022)	85.67	83.37	84.50
DSpERT†	87.37	87.93	**87.65**
DSpERT‡	87.00	87.33	$87.16_{\pm 0.50}$

Table 1: Results of English nested entity recognition. * means that the model is trained with both the training and development splits. † means the best score; ‡ means the average score of multiple independent runs; the subscript number is the corresponding standard deviation.

CoNLL 2003			
Model	Prec.	Rec.	F1
Peters et al. (2018)‡	–	–	$92.22_{\pm 0.10}$
Devlin et al. (2019)	–	–	92.8
Li et al. (2020b)	92.33	94.61	93.04
Yu et al. (2020)*	93.7	93.3	93.5
Yan et al. (2021)*	92.61	93.87	93.24
Li et al. (2022)‡	92.71	93.44	93.07
Zhu and Li (2022)	93.61	93.68	93.65
Shen et al. (2022)*	93.29	92.46	92.87
DSpERT†	93.48	93.93	**93.70**
DSpERT‡	93.39	93.88	$93.64_{\pm 0.06}$

OntoNotes 5			
Model	Prec.	Rec.	F1
Li et al. (2020b)	92.98	89.95	91.11
Yu et al. (2020)	91.1	91.5	91.3
Yan et al. (2021)	89.99	90.77	90.38
Li et al. (2022)‡	90.03	90.97	90.50
Zhu and Li (2022)	91.75	91.74	91.74
Shen et al. (2022)	91.43	90.73	90.96
DSpERT†	91.46	92.05	**91.76**
DSpERT‡	90.87	91.25	$91.06_{\pm 0.26}$

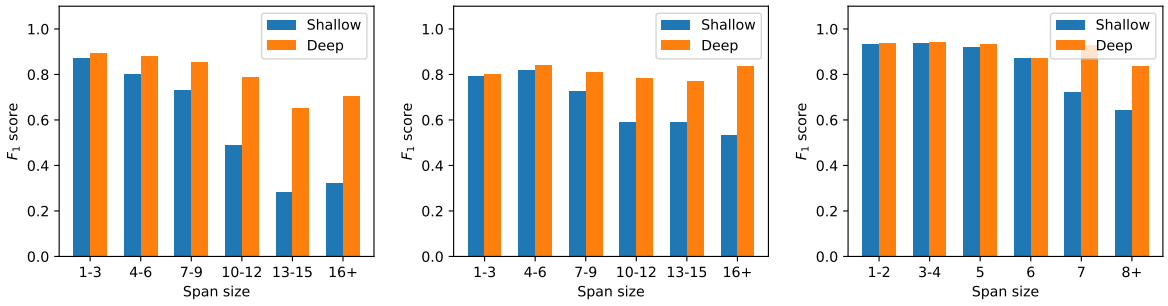
Table 2: Results of English flat entity recognition. * means that the model is trained with both the training and development splits. † means the best score; ‡ means the average score of multiple independent runs; the subscript number is the corresponding standard deviation.

Depth	ACE04	GENIA	CoNLL03
Shallow agg. (i.e., depth = 0)			
w/ max-pooling	$82.22_{\pm 0.64}$	$79.44_{\pm 0.20}$	$92.99_{\pm 0.32}$
w/ mean-pooling	$80.90_{\pm 0.28}$	$73.83_{\pm 0.42}$	$91.97_{\pm 0.14}$
w/ mul. attention	$84.38_{\pm 0.58}$	$76.54_{\pm 2.70}$	$93.21_{\pm 0.16}$
w/ add. attention	$83.73_{\pm 0.52}$	$76.23_{\pm 3.27}$	$93.05_{\pm 0.04}$
DSpERT			
depth = 2	$87.87_{\pm 0.13}$	$80.66_{\pm 0.36}$	$93.30_{\pm 0.09}$
depth = 4	$87.88_{\pm 0.41}$	$80.88_{\pm 0.39}$	$93.38_{\pm 0.08}$
depth = 6	$87.81_{\pm 0.13}$	$81.01_{\pm 0.22}$	$93.40_{\pm 0.13}$
depth = 8	$88.00_{\pm 0.22}$	$81.13_{\pm 0.25}$	$93.48_{\pm 0.08}$
depth = 10	$88.00_{\pm 0.21}$	$81.12_{\pm 0.14}$	$93.51_{\pm 0.10}$
depth = 12	$\mathbf{88.05}_{\pm 0.18}$	$\mathbf{81.46}_{\pm 0.25}$	$\mathbf{93.64}_{\pm 0.06}$

Table 3: The effect of depth. The underlined specification is the one used in our main experiments. All the results are average scores of five independent runs, with subscript standard deviations.

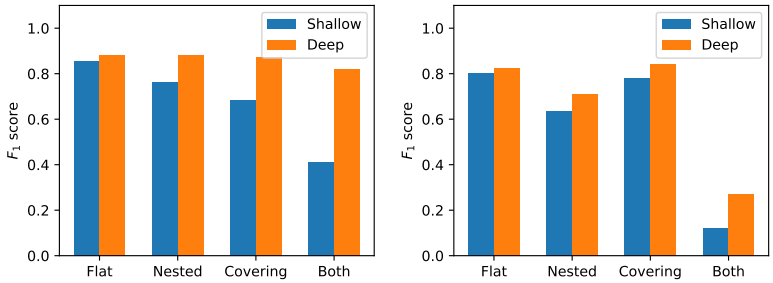
where the span representations are aggregated from the top token representations by max-pooling, mean-pooling, multiplicative attention or additive attention (See Appendix A for details). All the models are trained with the same recipe used in our main experiments. It shows that the 12-layer deep span representations achieve higher performance than its shallow counterparts equipped with any potential aggregating function, across all datasets.

We further run DSpERT with $\tilde{L}$ ($\tilde{L} < L$) span Transformer blocks, where the initial aggregation happens at the $(L - \tilde{L})$-th layer and the span Transformer corresponds to the top/last $\tilde{L}$ Transformer blocks. These models may be thought of as intermediate configurations between fully deep span representations and fully shallow ones. As displayed in Table 3, the F_1 score in general experiences a monotonically increasing trend when depth



(a) ACE 2004 (b) GENIA (c) CoNLL 2003

Figure 2: F_1 scores on spans of different lengths. All the results are average scores of five independent runs.



(a) ACE 2004 (b) GENIA

Figure 3: F_1 scores on spans with different nested structures. "Nested" means the spans that are nested inside a ground-truth entity which covers other ground-truth entities; "Covering" means the spans that cover a ground-truth entity which is nested inside other ground-truth entities; "Both" means the spans that are both "Nested" and "Covering"; "Flat" means the spans that are neither "Nested" nor "Covering". All the results are average scores of five independent runs.

$\tilde{L}$ increases from 2 to 12; this pattern holds for all three datasets. These results further strengthen our argument that the depth positively contributes to the quality of span representations.

Appendix F provides extensive ablation studies evaluating other components.

4.4 Effect on Long-Span Entities

The recognition of long-span entities is a long-tail and challenging problem. Taking ACE 2004 as an example, the ground-truth entities longer than 10 tokens only account for 2.8%, and the maximum length reaches 57. Empirical evidence also illustrates that existing NER models show relatively weak performance on long entities (e.g., Shen et al., 2021; Yuan et al., 2022).

Figure 2 presents the F_1 scores grouped by different span lengths. In general, the models based on shallow span representations perform relatively well on the short entities, but struggle for the long ones. However, DSpERT show much higher F_1 scores on the long entities, without any performance sacrifice on the short ones. For ACE 2004, DSpERT outperforms its shallow counterpart by 2%–12% absolute F_1 score on spans shorter than 10, while this difference exceeds 30% for spans longer than 10. Similar patterns are observed on GENIA and CoNLL 2003.

Conceptually, a longer span contains more information, so it would be more difficult to be encoded into a fixed-length vector, i.e., the span representation. According to our experimental results, the shallow aggregation fails to fully preserve the semantics in the original token representations, especially for long spans. The DSpERT, however, allows complicated interactions between tokens through multiple layers; in particular, longer spans experience more interactions. This mechanism amplifies the performance gain on long entities.

4.5 Effect on Nested Structures

Even in nested NER datasets, the nested entities are less than the flat ones (See Table 5). To deliberately investigate the performance on nested entities, we look into two subsets of spans that are directly related to nested structures: (1) *Nested*: the spans

that are nested inside a ground-truth entity which covers other ground-truth entities; (2) *Covering*: the spans that cover a ground-truth entity which is nested inside other ground-truth entities.

Figure 3 depicts the F_1 scores grouped by different nested structures. Consistent to a common expectation, nested structures create significant difficulties for entity recognition. Compared to the flat ones, a shallow span-based model encounters a substantial performance degradation on the nested structures, especially on the spans in both the *Nested* and *Covering* subsets. On the other hand, our deep span representations perform much well. For ACE 2004, DSpERT presents 2% higher absolute F_1 score than the shallow model on flat spans, but achieves about 40% higher score on spans in both the *Nested* and *Covering* subsets. The experiments on GENIA report similar results, although the difference in performance gain becomes less substantial.

As previously emphasized, shallowly aggregated span representations are technically coupled if the spans overlap. This explains why such models perform poorly on nested structures. Our model addresses this issue by deep and multi-layered construction of span representations. Implied by the experimental results, deep span representations are less coupled and more easily separable for overlapping spans.

5 Analysis of Pre-Logit Representations

5.1 ℓ_2-Norm and Cosine Similarity

We calculate the ℓ_2-norm and cosine similarity of the span representations, i.e., $\boldsymbol{z}_{ij}$ in Eq. (6). As presented in Table 4, the deep span representations have larger ℓ_2-norm than those of the shallow counterpart. Although the variance of representations inevitably shrinks during the aggregating process from the perspective of statistics, this result implies that deep span representations are less restricted and thus able to flexibly represent rich semantics. In addition, the deep span representations are associated with higher within-class similarities and lower between-class similarities, suggesting that the representations are more tightly clustered within each category, but more separable between categories. Apparently, this nature contributes to the high classification performance.

We further investigate the pre-logit weight, i.e., $\mathbf{W}$ in Eq. (6). First, the trained DSpERT has a pre-logit weight with a smaller ℓ_2-norm. According to a common understanding of neural networks, smaller norm implies that the model is simpler and thus more generalizable.

Second, as indicated by Müller et al. (2019), a typical neural classification head can be regarded as a *template-matching* mechanism, where each row vector of $\mathbf{W}$ is a *template*.[7] Under this interpretation, each template "stands for" the overall direction of the span representations of the corresponding category in the feature space. As shown in Table 4, the absolute cosine values between the templates of DSpERT are fairly small. In other words, the templates are approximately orthogonal, which suggests that different entity categories are uncorrelated and separately occupy distinctive subareas in the feature space. This pattern, however, is not present for the shallow models.

5.2 Visualization

Figure 4 visualizes the span representations dimension-reduced by principal component analysis (PCA). The results are quite impressive. The representations by shallow aggregation are scattered over the plane. Although they are largely clustered by categories, the boundaries are mixed and ambiguous. In contrast, the deep span representations group by relatively clear and tight clusters, corresponding to the ground-truth categories. Except for some outliers, each pair of the clusters can be easily separable in this projected plane. This is also consistent to the aforementioned finding that deep span representations have high within-class similarities but low between-class similarities.

6 Discussion and Conclusion

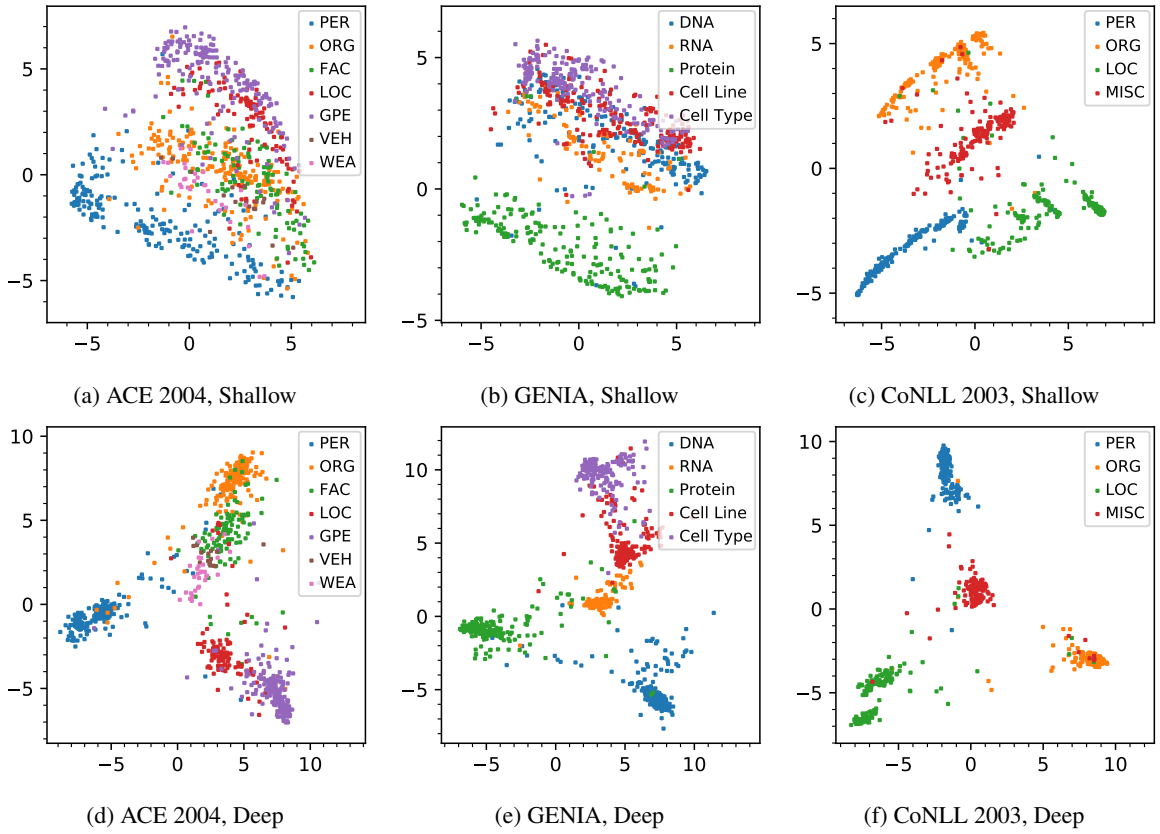
Neural NLP models have been rigidly adhere to the paradigm where an encoder produces token-level representations, and a task-specific decoder receives these representations, computes the loss and yields the outputs (Collobert et al., 2011). This design works well on most, if not all, NLP tasks; and it may also deserve a credit for facilitating NLP pretraining (Peters et al., 2018; Devlin et al., 2019), since such common structure in advance bridges the pretraining and downstream phases.

However, this paradigm may be sub-optimal for specific tasks. In span-based NER (or information extraction from a broader perspective), the smallest

[7] Conceptually, if vector $\boldsymbol{z}_{ij}$ is most matched/correlated with the k-th row vector of $\mathbf{W}$, then the k-th logit will be most activated. Refer to Müller et al. (2019) for more details.

	ACE 2004		GENIA		CoNLL 2003	
	Shallow	Deep	Shallow	Deep	Shallow	Deep
Pre-logit representations						
ℓ_2-norm	$9.58_{\pm 0.17}$	$14.55_{\pm 0.33}\uparrow$	$9.89_{\pm 0.18}$	$11.69_{\pm 0.52}\uparrow$	$9.08_{\pm 0.17}$	$13.49_{\pm 0.50}\uparrow$
Cosine within pos class	$0.70_{\pm 0.02}$	$0.80_{\pm 0.01}\uparrow$	$0.78_{\pm 0.01}$	$0.83_{\pm 0.01}\uparrow$	$0.79_{\pm 0.01}$	$0.88_{\pm 0.00}\uparrow$
Cosine between pos vs. pos	$0.45_{\pm 0.02}$	$0.37_{\pm 0.01}\downarrow$	$0.54_{\pm 0.02}$	$0.29_{\pm 0.01}\downarrow$	$0.35_{\pm 0.02}$	$0.32_{\pm 0.01}\downarrow$
Cosine between pos vs. neg	$0.48_{\pm 0.03}$	$0.35_{\pm 0.03}\downarrow$	$0.55_{\pm 0.01}$	$0.35_{\pm 0.02}\downarrow$	$0.39_{\pm 0.01}$	$0.37_{\pm 0.02}\downarrow$
Pre-logit weight/templates **W**						
ℓ_2-norm	$1.93_{\pm 0.28}$	$1.57_{\pm 0.03}\downarrow$	$1.84_{\pm 0.08}$	$1.39_{\pm 0.02}\downarrow$	$1.71_{\pm 0.04}$	$1.39_{\pm 0.01}\downarrow$
Abs cosine	$0.17_{\pm 0.12}$	$0.10_{\pm 0.06}\downarrow$	$0.31_{\pm 0.11}$	$0.10_{\pm 0.10}\downarrow$	$0.23_{\pm 0.08}$	$0.05_{\pm 0.03}\downarrow$
Abs cosine between pos vs. pos	$0.13_{\pm 0.10}$	$0.10_{\pm 0.07}\downarrow$	$0.27_{\pm 0.11}$	$0.10_{\pm 0.11}\downarrow$	$0.17_{\pm 0.04}$	$0.05_{\pm 0.04}\downarrow$
Abs cosine between pos vs. neg	$0.31_{\pm 0.06}$	$0.10_{\pm 0.05}\downarrow$	$0.40_{\pm 0.06}$	$0.09_{\pm 0.04}\downarrow$	$0.32_{\pm 0.04}$	$0.05_{\pm 0.03}\downarrow$

Table 4: ℓ_2-norm and cosine similarity of pre-logit representations and templates. "pos" means the positive types (i.e., entity types); "neg" means the negative type (i.e., non-entity type). ↑/↓ indicates that DSpERT presents a metric higher/lower than its shallow counterpart. All the metrics are first averaged within each experiment, and then averaged over five independent experiments, reported with subscript standard deviations.



(a) ACE 2004, Shallow (b) GENIA, Shallow (c) CoNLL 2003, Shallow

(d) ACE 2004, Deep (e) GENIA, Deep (f) CoNLL 2003, Deep

Figure 4: PCA visualization of pre-logit span representations of entities in the testing set.

modeling unit should be spans instead of tokens; and thus the span representations should be crucial. This originally motivates our DSpERT. In addition, DSpERT also successfully shows how to exploit the pretrained weights beyond the original Transformer structure, i.e., adapting the weights from computing token representations for span representations. We believe that adding span representation learning in the pretraining stage will further contribute positively.

In conclusion, deep and span-specific representations can significantly boost span-based neural NER models. Our DSpERT achieves SOTA results on six well-known NER benchmarks; the model presents pronounced effect on long-span entities and nested structures. Further analysis shows that the resulting deep span representations are well structured and easily separable in the feature space.

7 Limitations

To some extent, DSpERT pursues performance and interpretability over computational efficiency. The major computational cost of a Transformer encoder is on the multihead attention module and FFN. For a T-length input and a d-dimensional Transformer encoder, the per-layer complexities of the multihead attention and FFN are of order $O(T^2d)$ and $O(Td^2)$, respectively. When the maximum span size $K \ll T$, our span Transformer brings additional $O(K^2Td)$ complexity on the attention module, and $O(KTd^2)$ complexity on the FFN. Empirically, training a DSpERT consumes about five times the time for a shallow model of a same scale. However, this issue can be mitigated if we use fewer layers for the span Transformer (Subsection 4.3).

As noted, we empirically choose the maximum span size K such that it covers most entities in the training and development splits. From the perspective of F_1 score, this heuristic works well, and DSpERT performs favourably on long-span entities as long as they are covered. However, the entities with extreme lengths beyond K will be theoretically irretrievable.

References

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(ARTICLE):2493–2537.

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.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of 5th International Conference on Learning Representations*. OpenReview.net.

Markus Eberts and Adrian Ulges. 2020. Span-based joint entity and relation extraction with Transformer pre-training. In *Proceedings of the 24th European Conference on Artificial Intelligence*, Santiago de Compostela, Spain.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Heng Ji, Xiaoman Pan, Boliang Zhang, Joel Nothman, James Mayfield, Paul McNamee, and Cash Costello. 2017. Overview of TAC-KBP2017 13 languages entity discovery and linking. In *Proceedings of the 2017 Text Analysis Conference*, Gaithersburg, Maryland, USA. NIST.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459, New Orleans, Louisiana. Association for Computational Linguistics.

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, Volume 1 (Long Papers)*, pages 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus – a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. 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*, pages 260–270, San Diego, California. 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 *Proceedings of the AAAI Conference on Artificial Intelligence*.

Xiaonan Li, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020a. FLAT: Chinese NER using flat-lattice transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6836–6842, Online. Association for Computational Linguistics.

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

Yangming Li, Shuming Shi, et al. 2020c. Empirical analysis of unlabeled entity problem in named entity recognition. In *International Conference on Learning Representations*.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5182–5192, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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*, pages 857–867, Lisbon, Portugal. Association for Computational Linguistics.

Ruotian Ma, Minlong Peng, Qi Zhang, Zhongyu Wei, and Xuanjing Huang. 2020. Simplify the usage of lexicon in Chinese NER. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5951–5960, Online. Association for Computational Linguistics.

Xuezhe Ma and Eduard 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 (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Hiroki Ouchi, Jun Suzuki, Sosuke Kobayashi, Sho Yokoi, Tatsuki Kuribayashi, Ryuto Konno, and Kentaro Inui. 2020. Instance-based learning of span representations: A case study through named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6452–6459, Online. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318. PMLR.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for Chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554, Lisbon, Portugal. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. 2021. Locate and label: A two-stage identifier for nested named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2782–2794, Online. Association for Computational Linguistics.

Yongliang Shen, Xiaobin Wang, Zeqi Tan, Guangwei Xu, Pengjun Xie, Fei Huang, Weiming Lu, and Yueting Zhuang. 2022. Parallel instance query network for named entity recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 947–961, Dublin, Ireland. Association for Computational Linguistics.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849, Brussels, Belgium. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Shuang Wu, Xiaoning Song, and Zhenhua Feng. 2021. MECT: Multi-metadata embedding based cross-transformer for Chinese named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1529–1539, Online. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 5808–5822, Online. 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*, pages 6470–6476, Online. 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*, pages 3174–3186, Dublin, Ireland. Association for Computational Linguistics.

Yue Zhang and Jie Yang. 2018. Chinese NER using lattice LSTM. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564, Melbourne, Australia. 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)*, pages 7096–7108, Dublin, Ireland. Association for Computational Linguistics.

A (Shallowly) Aggregating Functions

Given token representations $\mathbf{H} \in \mathbb{R}^{T \times d}$, a model can shallowly aggregate them in corresponding positions to construct span representations. Formally, the span representation of (i, j) can be built by:

Max-pooling. Applying max-pooling to $\mathbf{H}_{[i:j]}$ over the first dimension.

Mean-pooling. Applying mean-pooling to $\mathbf{H}_{[i:j]}$ over the first dimension.

Multiplicative Attention. Computing

$$\mathrm{softmax}\left(\boldsymbol{u}^{\mathsf{T}} \tanh\left(\mathbf{W}\mathbf{H}_{[i:j]}^{\mathsf{T}}\right)\right)\mathbf{H}_{[i:j]},$$

where $\mathbf{W}$ and $\boldsymbol{u}$ are learnable parameters.

Additive Attention. Computing

$$\mathrm{softmax}\left(\boldsymbol{u}^{\mathsf{T}} \tanh\left(\mathbf{W}\left(\mathbf{H}_{[i:j]} \oplus \boldsymbol{v}\right)^{\mathsf{T}}\right)\right)\mathbf{H}_{[i:j]},$$

where $\mathbf{W}$, $\boldsymbol{u}$ and $\boldsymbol{v}$ are learnable parameters; $\oplus$ means concatenation over the second dimension, and vector $\boldsymbol{v}$ should be repeated for $j - i$ times before the concatenation.

In general, either multiplicative or additive attention computes normalized weights over the sequence dimension of span (i, j), where the weights are dependent on the values of $\mathbf{H}_{[i:j]}$; and then applies the weights to $\mathbf{H}_{[i:j]}$, resulting in weighted average values.

B Datasets

ACE 2004 and ACE 2005 are two English nested NER datasets, either of which contains seven entity types, i.e., Person, Organization, Facility, Location, Geo-political Entity, Vehicle, Weapon. Our data processing and splits follow Lu and Roth (2015).

GENIA (Kim et al., 2003) is a nested NER corpus on English biological articles. Our data processing follows Lu and Roth (2015), resulting in five entity types (DNA, RNA, Protein, Cell line, Cell type); data splits follow Yan et al. (2021) and Li et al. (2022).

KBP 2017 (Ji et al., 2017) is an English nested NER corpus including text from news, discussion forum, web blog, tweets and scientific literature. It contains five entity categories, i.e., Person, Geo-political Entity, Organization, Location, and Facility. Our data processing and splits follow Lin et al. (2019) and Shen et al. (2022).

		ACE04	ACE05	GENIA	KBP17	CoNLL03	OntoNotes 5
#Sent.	All	8,507	9,341	18,546	15,358	22,137	76,714
	Train	6,799	7,336	15,023	10,546	14,987	59,924
	Dev.	829	958	1,669	545	3,466	8,528
	Test	879	1,047	1,854	4,267	3,684	8,262
#Type		7	7	5	5	4	18
#Token		173,796	176,575	471,264	300,345	302,811	1,388,955
#Entity		27,749	30,931	56,046	45,714	35,089	104,151
#Nested		7,832	7,460	5,431	7,479	–	–

Table 5: Descriptive statistics of datasets. #Sent. denotes the number of sentences; #Type denotes the number of entity types; #Token denotes the number of tokens; #Entity denotes the number of entities; #Nested denotes the number of entities that are nested in other entities.

	ACE04	ACE05	GENIA	KBP17	CoNLL03	OntoNotes 5
PLM	RoBERTa-base					
Maximum span size	25	25	18	16	10	16
Initial aggregation	Max	Max	Max	Max	Max	Max
LSTM hidden size	400	400	400	400	400	400
LSTM layers	1	1	1	1	1	1
FFN hidden size	300	300	300	300	300	300
FFN layers	1	1	1	1	1	1
Boundary smoothing ε	0.1	0.1	0.1	0.1	0.1	0.1
Number of epochs	50	50	30	50	50	30
Learning rate						
Pretrained weights	2e-5	2e-5	4e-5	2e-5	2e-5	2e-5
Other weights	2e-3	2e-3	2e-3	2e-3	2e-3	2e-3
Batch size	48	48	16	48	48	16
Number of parameters (M)	212.9 (w/o weight sharing); or 126.3 (w/ weight sharing)					
Training time (hours on A6000)	13.9	15.0	8.3	16.2	3.3	15.3

Table 6: Hyperparameters and computational costs of main experiments.

CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) is an English flat NER benchmark with four entity types, i.e., Person, Organization, Location and Miscellaneous. We use the original data splits for experiments.

OntoNotes 5 is a large-scale English flat NER benchmark, which has 18 entity types. Our data processing and splits follow Pradhan et al. (2013).

Table 5 presents the descriptive statistics of the datasets.

C Implementation Details

Hyperparameters. We choose RoBERTa (Liu et al., 2019) as the PLM to initialize the weights in the Transformer blocks and span Transformer blocks. The PLMs used in our main experiments are all of the `base` size (768 hidden size, 12 layers).

For span Transformer blocks, the maximum span size K is specifically determined for each dataset. In general, a larger K would improve the recall performance (entities longer than K will never be recalled), but significantly increase the computation cost. We empirically choose K such that it covers most entities in the training and development splits. For example, most entities are short in CoNLL 2003, so we use $K = 10$; while entities are relatively long in ACE 2004 and ACE 2005, so we use $K = 25$. We use max-pooling as the initial aggregating function.

We find it beneficial to additionally include a BiLSTM (Hochreiter and Schmidhuber, 1997) before passing the span representations to the entity classifier. The BiLSTM has one layer and 400 hidden states, with dropout rate of 0.5. In the entity classifier, the FFN has one layer and 300 hidden states, with dropout rate of 0.4, and the activation is ReLU (Krizhevsky et al., 2012). In addition, boundary smoothing (Zhu and Li, 2022) with $\varepsilon = 0.1$ is applied to loss computation.

We train the models by the AdamW optimizer (Loshchilov and Hutter, 2018) for 50 epochs with the batch size of 48. Gradients are clipped at ℓ_2-norm of 5 (Pascanu et al., 2013). The learning rates are 2e-5 and 2e-3 for pretrained weights and randomly initialized weights, respectively; a sched-

uler of linear warmup is applied in the first 20% steps followed by linear decay. For some datasets, a few hyperparameters are further tuned and thus slightly different from the above ones.

The experiments are run on NVIDIA RTX A6000 GPUs. More details on the hyperparameters and computational costs are reported in Table 6.

Evaluation. An entity is considered correctly recognized if its predicted type and boundaries exactly match the ground truth.

The model checkpoint with the best F_1 score throughout the training process on the development split is used for evaluation. The evaluation metrics are micro precision rate, recall rate and F_1 score on the testing split. Unless otherwise noted, we run each experiment for five times and report the average metrics with corresponding standard deviations.

D Categorical Results

Table 7 lists the category-specific results on ACE 2004, GENIA and CoNLL 2003. As a strong baseline, the classic biaffine model (Yu et al., 2020) is re-implemented with PLM and hyperparameters consistent with our DSpERT; note that our re-implementation achieves higher performances than the scores reported in the original paper. The categorical results show that DSpERT outperforms the biaffine model across almost all the categories of the three datasets, except for Geo-political Entity and Vehicle from ACE 2004.

E Results on Chinese NER

Table 8 shows the experimental results on two Chinese flat NER datasets: Weibo NER (Peng and Dredze, 2015) and Resume NER (Zhang and Yang, 2018). DSpERT achieves 72.64% and 96.72% best F_1 scores on the two benchmarks; they are also quite close to the recently reported SOTA results.

F Additional Ablation Studies

Weight Sharing. As described, the span Transformer shares the same structure with the Transformer, but their weights are independent and separately initialized from the PLM. A straightforward idea is to tie the corresponding weights between these two modules, which can reduce the model parameters and conceptually performs as a regularization technique.

	ACE 2004		
	Biaffine	DSpERT	Δ
PER	$91.35_{\pm 0.10}$	$91.50_{\pm 0.16}$	+0.14
ORG	$82.49_{\pm 0.54}$	$83.59_{\pm 0.33}$	+1.10
FAC	$71.18_{\pm 1.30}$	$72.28_{\pm 1.43}$	+1.10
LOC	$73.38_{\pm 1.55}$	$75.22_{\pm 0.86}$	+1.84
GPE	$89.30_{\pm 0.36}$	$89.16_{\pm 0.44}$	-0.15
VEH	$87.18_{\pm 2.66}$	$84.35_{\pm 3.58}$	-2.83
WEA	$70.87_{\pm 4.18}$	$79.88_{\pm 1.41}$	+9.00
Overall	$87.64_{\pm 0.19}$	$\mathbf{88.05}_{\pm 0.18}$	+0.41
	GENIA		
	Biaffine	DSpERT	Δ
DNA	$77.50_{\pm 0.44}$	$77.75_{\pm 0.56}$	+0.26
RNA	$83.75_{\pm 0.68}$	$85.02_{\pm 1.31}$	+1.27
Protein	$84.00_{\pm 0.36}$	$84.57_{\pm 0.26}$	+0.57
Cell Line	$74.60_{\pm 0.48}$	$76.39_{\pm 0.96}$	+1.79
Cell Type	$75.89_{\pm 0.46}$	$76.10_{\pm 0.48}$	+0.22
Overall	$80.93_{\pm 0.24}$	$\mathbf{81.46}_{\pm 0.25}$	+0.53
	CoNLL 2003		
	Biaffine	DSpERT	Δ
PER	$96.74_{\pm 0.23}$	$96.94_{\pm 0.11}$	+0.20
ORG	$92.92_{\pm 0.13}$	$93.09_{\pm 0.14}$	+0.17
LOC	$94.83_{\pm 0.18}$	$94.96_{\pm 0.13}$	+0.12
MISC	$83.83_{\pm 0.28}$	$84.52_{\pm 0.56}$	+0.69
Overall	$93.37_{\pm 0.10}$	$\mathbf{93.64}_{\pm 0.06}$	+0.27

Table 7: Categorical F_1 scores by biaffine model and DSpERT. All the results are average scores of five independent runs, with subscript standard deviations.

Table 9 presents the results. Sharing the weights results in higher F_1 score on ACE 2004, but lower results on GENIA and CoNLL 2003; the performance differences are largely insignificant. In addition to the performance, weight sharing has very limited effect on reducing training and inference cost — both the forward and backward computations remain almost unchanged. However, it does effectively halve the parameter number; hence, weight sharing would be a preferred option when the storage space is limited.

Initial Aggregation Functions. We test different functions for initial aggregation in the span Transformer. As shown in Table 10, max-pooling outperforms all other alternatives, although the performance differences are limited. This result is different from that in the shallow setting, where max-pooling underperforms multiplicative and additive attentions (Table 3).

One possible explanation is that, the span Transformer blocks with weights initialized from the PLM have already performed very sophisticated

Weibo NER			
Model	Prec.	Rec.	F1
Ma et al. (2020)	–	–	70.50
Li et al. (2020a)	–	–	68.55
Wu et al. (2021)	–	–	70.43
Li et al. (2022)‡	70.84	73.87	72.32
Zhu and Li (2022)	70.16	75.36	**72.66**
DSpERT†	74.56	70.81	72.64
DSpERT‡	71.09	71.58	$71.30_{\pm 0.86}$
Resume NER			
Model	Prec.	Rec.	F1
Ma et al. (2020)	96.08	96.13	96.11
Li et al. (2020a)	–	–	95.86
Wu et al. (2021)	–	–	95.98
Li et al. (2022)‡	96.96	96.35	96.65
Zhu and Li (2022)	96.63	96.69	96.66
DSpERT†	96.69	96.75	**96.72**
DSpERT‡	96.44	96.58	$96.51_{\pm 0.17}$

Table 8: Results of Chinese flat entity recognition. † means the best score; ‡ means the average score of multiple independent runs; the subscript number is the corresponding standard deviation.

Weight Sharing	ACE04	GENIA	CoNLL03
×	$88.05_{\pm 0.18}$	$\mathbf{81.46}_{\pm 0.25}$	$\mathbf{93.64}_{\pm 0.06}$
✓	$\mathbf{88.11}_{\pm 0.23}$	$81.19_{\pm 0.26}$	$93.43_{\pm 0.17}$

Table 9: The effect of weight sharing. The underlined specification is the one used in our main experiments. All the results are average scores of five independent runs, with subscript standard deviations.

multi-head attentions through multiple layers, so one extra attention layer with randomly initialized weights cannot take positive effect further, or even plagues the model. Max-pooling, on the other hand, is parameter-free and thus more advantageous in this case.

Pretrained Language Models. Table 10 also lists the results by alternative PLMs. In general, BERT-base and BERT-large (Devlin et al., 2019) underperforms RoBERTa-base (Liu et al., 2019), while RoBERTa-large can further improve the performance by 0.54, 0.52 and 0.06 percentage F_1 scores on ACE 2004, GENIA and CoNLL 2003, respectively. These results are consistent with Zhu and Li (2022), confirming the superiority of RoBERTa in NER tasks.

Since GENIA is a biological corpus, some previous studies use BioBERT on this benchmark (Shen et al., 2021, 2022). We also test BioBERT with DSpERT on GENIA. The results

	ACE04	GENIA	CoNLL03
Initial agg.			
w/ max-pooling	$88.05_{\pm 0.18}$	$81.46_{\pm 0.25}$	$93.64_{\pm 0.06}$
w/ mean-pooling	$87.77_{\pm 0.31}$	$81.31_{\pm 0.14}$	$93.58_{\pm 0.11}$
w/ mul. attention	$87.67_{\pm 0.29}$	$81.36_{\pm 0.17}$	$93.55_{\pm 0.09}$
w/ add. attention	$87.90_{\pm 0.11}$	$81.20_{\pm 0.12}$	$93.56_{\pm 0.12}$
PLM			
w/ RoBERTa-b	$88.05_{\pm 0.18}$	$81.46_{\pm 0.25}$	$93.64_{\pm 0.06}$
w/ RoBERTa-l	$\mathbf{88.59}_{\pm 0.27}$	$\mathbf{81.98}_{\pm 0.41}$	$\mathbf{93.70}_{\pm 0.12}$
w/ BERT-b	$86.38_{\pm 0.20}$	$79.13_{\pm 0.16}$	$91.90_{\pm 0.08}$
w/ BERT-l	$87.73_{\pm 0.30}$	$79.27_{\pm 0.20}$	$92.79_{\pm 0.14}$
w/ BioBERT-b		$81.52_{\pm 0.26}$	
w/ BioBERT-l		$81.78_{\pm 0.26}$	
Others			
w/o BiLSTM	$87.71_{\pm 0.05}$	$81.29_{\pm 0.33}$	$93.42_{\pm 0.11}$
w/o BS	$87.66_{\pm 0.26}$	$81.02_{\pm 0.27}$	$93.45_{\pm 0.17}$

Table 10: Results of ablation studies. "b" and "l" mean the PLM sizes of `base` and `large`, respectively; for `large` PLM, span Transformer has 12 layers. "BS" means boundary smoothing. The underlined specification is the one used in our main experiments. All the results are average scores of five independent runs, with subscript standard deviations.

show that BioBERT can achieve performance competitive to RoBERTa.

BiLSTM and Boundary Smoothing. As presented in Table 10, removing the BiLSTM layer will result in a drop of 0.2–0.4 percentage F_1 scores. In addition, replacing boundary smoothing (Zhu and Li, 2022) with the standard cross entropy loss will reduce the F_1 scores by similar magnitudes.