TENER: Adapting Transformer Encoder for Named Entity Recognition

Hang Yan, Bocao Deng, Xiaonan Li, Xipeng Qiu[*]

School of Computer Science, Fudan University
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
{hyan19,xpqiu}@fudan.edu.cn, {dengbocao,lixiaonan1208}@gmail.com

这是一条gzq的注释笔记。

Abstract

Bidirectional long short-term memory networks (BiLSTMs) have been widely used as an encoder for named entity recognition (NER) task. Recently, the fully-connected self-attention architecture (aka Transformer) is broadly adopted in various natural language processing (NLP) tasks owing to its parallelism and advantage in modeling the long-range context. Nevertheless, the performance of the vanilla Transformer in NER is not as good as it is in other NLP tasks. In this paper, we propose TENER, a NER architecture adopting adapted Transformer Encoder to model the character-level features and word-level features. By incorporating the direction-aware, distance-aware and un-scaled attention, we prove the Transformer-like encoder is just as effective for NER as other NLP tasks. Experiments on six NER datasets show that TENER achieves superior performance than the prevailing BiLSTM-based models.

1 Introduction

The named entity recognition (NER) is the task of finding the start and end of an entity in a sentence and assigning a class for this entity. NER has been widely studied in the field of natural language processing (NLP) because of its potential assistance in question generation (Zhou et al., 2017), relation extraction (Miwa and Bansal, 2016), and coreference resolution (Fragkou, 2017). Since (Collobert et al., 2011), various neural models have been introduced to avoid hand-crafted features (Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016).

NER is usually viewed as a sequence labeling task, the neural models usually contain three components: word embedding layer, context encoder layer, and decoder layer (Huang et al., 2015; Ma

and Hovy, 2016; Lample et al., 2016; Chiu and Nichols, 2016; Chen et al., 2019; Zhang et al., 2018; Gui et al., 2019b). The difference between various NER models mainly lies in the variance in these components.

Recurrent Neural Networks (RNNs) are widely employed in NLP tasks due to its sequential characteristic, which is aligned well with language. Specifically, bidirectional long short-term memory networks (BiLSTM) (Hochreiter and Schmidhuber, 1997) is one of the most widely used RNN structures. (Huang et al., 2015) was the first one to apply the BiLSTM and Conditional Random Fields (CRF) (Lafferty et al., 2001) to sequence labeling tasks. Owing to BiLSTM's high power to learn the contextual representation of words, it has been adopted by the majority of NER models as the encoder (Ma and Hovy, 2016; Lample et al., 2016; Zhang et al., 2018; Gui et al., 2019b).

Recently, Transformer (Vaswani et al., 2017) began to prevail in various NLP tasks, like machine translation (Vaswani et al., 2017), language modeling (Radford et al., 2018), and pretraining models (Devlin et al., 2018). The Transformer encoder adopts a fully-connected self-attention structure to model the long-range context, which is the weakness of RNNs. Moreover, Transformer has better parallelism ability than RNNs. However, in the NER task, Transformer encoder has been reported to perform poorly (Guo et al., 2019), our experiments also confirm this result. Therefore, it is intriguing to explore the reason why Transformer does not work well in NER task.

In this paper, we analyze the properties of Transformer and propose two specific improvements for NER.

The first is that the sinusoidal position embedding used in the vanilla Transformer is aware of distance but unaware of the directionality. In addition, this property will lose when used in the

---

[*]Corresponding author.

Figure 1: An example for NER. The relative direction is important in the NER task, because words before "Inc." are mostly to be an organization, words after "in" are more likely to be time or location. Besides, the distance between words is also important, since only continuous words can form an entity, the former "Louis Vuitton" can not form an entity with the "Inc.".

vanilla Transformer. However, both the direction and distance information are important in the NER task. For example in Fig 1, words after "in" are more likely to be a location or time than words before it, and words before "Inc." are mostly likely to be of the entity type "ORG". Besides, an entity is a continuous span of words. Therefore, the awareness of distance might help the word better recognizes its neighbor. To endow the Transformer with the ability of direction- and distance-awareness, we adopt the relative positional encoding (Shaw et al., 2018; Huang et al., 2019; Dai et al., 2019). instead of the absolute position encoding. We propose a revised relative positional encoding that uses fewer parameters and performs better.

The second is an empirical finding. The attention distribution of the vanilla Transformer is scaled and smooth. But for NER, a sparse attention is suitable since not all words are necessary to be attended. Given a current word, a few contextual words are enough to judge its label. The smooth attention could include some noisy information. Therefore, we abandon the scale factor of dot-production attention and use an un-scaled and sharp attention.

With the above improvements, we can greatly boost the performance of Transformer encoder for NER.

Other than only using Transformer to model the word-level context, we also tried to apply it as a character encoder to model word representation with character-level information. The previous work has proved that character encoder is necessary to capture the character-level features and alleviate the out-of-vocabulary (OOV) problem (Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016; Xin et al., 2018). In NER, CNN is commonly used as the character encoder. However, we argue that CNN is also not perfect for representing character-level information, be-

cause the receptive field of CNN is limited, and the kernel size of the CNN character encoder is usually 3, which means it cannot correctly recognize 2-gram or 4-gram patterns. Although we can deliberately design different kernels, CNN still cannot solve patterns with discontinuous characters, such as "un..ily" in "unhappily" and "unnecessarily". Instead, the Transformer-based character encoder shall not only fully make use of the concurrence power of GPUs, but also have the potentiality to recognize different n-grams and even discontinuous patterns. Therefore, in this paper, we also try to use Transformer as the character encoder, and we compare four kinds of character encoders.

In summary, to improve the performance of the Transformer-based model in the NER task, we explicitly utilize the directional relative positional encoding, reduce the number of parameters and sharp the attention distribution. After the adaptation, the performance raises a lot, making our model even performs better than BiLSTM based models. Furthermore, in the six NER datasets, we achieve state-of-the-art performance among models without considering the pre-trained language models or designed features.

2 Related Work

2.1 Neural Architecture for NER

Collobert et al. (2011) utilized the Multi-Layer Perceptron (MLP) and CNN to avoid using task-specific features to tackle different sequence labeling tasks, such as Chunking, Part-of-Speech (POS) and NER. In (Huang et al., 2015), BiLSTM-CRF was introduced to solve sequence labeling questions. Since then, the BiLSTM has been extensively used in the field of NER (Chiu and Nichols, 2016; Dong et al., 2016; Yang et al., 2018; Ma and Hovy, 2016).

Despite BiLSTM's great success in the NER task, it has to compute token representations one by one, which massively hinders full exploitation of GPU's parallelism. Therefore, CNN has been proposed by (Strubell et al., 2017; Gui et al., 2019a) to encode words concurrently. In order to enlarge the receptive field of CNNs, (Strubell et al., 2017) used iterative dilated CNNs (ID-CNN).

Since the word shape information, such as the capitalization and n-gram, is important in recognizing named entities, CNN and BiLSTM have been used to extract character-level informa-

tion (Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016; Strubell et al., 2017; Chen et al., 2019).

Almost all neural-based NER models used pre-trained word embeddings, like Word2vec and Glove (Pennington et al., 2014; Mikolov et al., 2013). And when contextual word embeddings are combined, the performance of NER models will boost a lot (Peters et al., 2017, 2018; Akbik et al., 2018). ELMo introduced by (Peters et al., 2018) used the CNN character encoder and BiLSTM language models to get contextualized word representations. Except for the BiLSTM based pre-trained models, BERT was based on Transformer (Devlin et al., 2018).

2.2 Transformer

Transformer was introduced by (Vaswani et al., 2017), which was mainly based on self-attention. It achieved great success in various NLP tasks. Since the self-attention mechanism used in the Transformer is unaware of positions, to avoid this shortage, position embeddings were used (Vaswani et al., 2017; Devlin et al., 2018). Instead of using the sinusoidal position embedding (Vaswani et al., 2017) and learned absolute position embedding, Shaw et al. (2018) argued that the distance between two tokens should be considered when calculating their attention score. Huang et al. (2019) reduced the computation complexity of relative positional encoding from $O(l^2 d)$ to $O(ld)$, where l is the length of sequences and d is the hidden size. Dai et al. (2019) derived a new form of relative positional encodings, so that the relative relation could be better considered.

2.2.1 Transformer Encoder Architecture

We first introduce the Transformer encoder proposed in (Vaswani et al., 2017). The Transformer encoder takes in an matrix $H \in \mathbb{R}^{l \times d}$, where l is the sequence length, d is the input dimension. Then three learnable matrix W_q, W_k, W_v are used to project H into different spaces. Usually, the matrix size of the three matrix are all $\mathbb{R}^{d \times d_k}$, where d_k is a hyper-parameter. After that, the scaled dot-product attention can be calculated by the following equations,

$$Q, K, V = HW_q, HW_k, HW_v, \quad (1)$$

$$A_{t,j} = Q_t K_j^T, \quad (2)$$

$$\text{Attn}(K, Q, V) = \text{softmax}(\frac{A}{\sqrt{d_k}})V, \quad (3)$$

where Q_t is the query vector of the tth token, j is the token the tth token attends. K_j is the key vector representation of the jth token. The softmax is along the last dimension. Instead of using one group of W_q, W_k, W_v, using several groups will enhance the ability of self-attention. When several groups are used, it is called multi-head self-attention, the calculation can be formulated as follows,

$$Q^{(h)}, K^{(h)}, V^{(h)} = HW_q^{(h)}, HW_k^{(h)}, HW_v^{(h)}, \quad (4)$$

$$head^{(h)} = \text{Attn}(Q^{(h)}, K^{(h)}, V^{(h)}), \quad (5)$$

$$\text{MultiHead}(H) = [head^{(1)}; ...; head^{(n)}]W_O, \quad (6)$$

where n is the number of heads, the superscript h represents the head index. $[head^{(1)}; ...; head^{(n)}]$ means concatenation in the last dimension. Usually $d_k \times n = d$, which means the output of $[head^{(1)}; ...; head^{(n)}]$ will be of size $\mathbb{R}^{l \times d}$. W_o is a learnable parameter, which is of size $\mathbb{R}^{d \times d}$.

The output of the multi-head attention will be further processed by the position-wise feed-forward networks, which can be represented as follows,

$$\text{FFN}(x) = max(0, xW_1 + b_1)W_2 + b_2, \quad (7)$$

where W_1, W_2, b_1, b_2 are learnable parameters, and $W_1 \in \mathbb{R}^{d \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d}$, $b_1 \in \mathbb{R}^{d_{ff}}$, $b_2 \in \mathbb{R}^d$. d_{ff} is a hyper-parameter. Other components of the Transformer encoder includes layer normalization and Residual connection, we use them the same as (Vaswani et al., 2017).

2.2.2 Position Embedding

The self-attention is not aware of the positions of different tokens, making it unable to capture the sequential characteristic of languages. In order to solve this problem, (Vaswani et al., 2017) suggested to use position embeddings generated by sinusoids of varying frequency. The tth token's position embedding can be represented by the following equations

$$PE_{t,2i} = \sin(t/10000^{2i/d}), \quad (8)$$

$$PE_{t,2i+1} = \cos(t/10000^{2i/d}), \quad (9)$$

where i is in the range of $[0, \frac{d}{2}]$, d is the input dimension. This sinusoid based position embedding makes Transformer have an ability to model the position of a token and the distance of each two tokens. For any fixed offset k, PE_{t+k} can be represented by a linear transformation of PE_t (Vaswani et al., 2017).
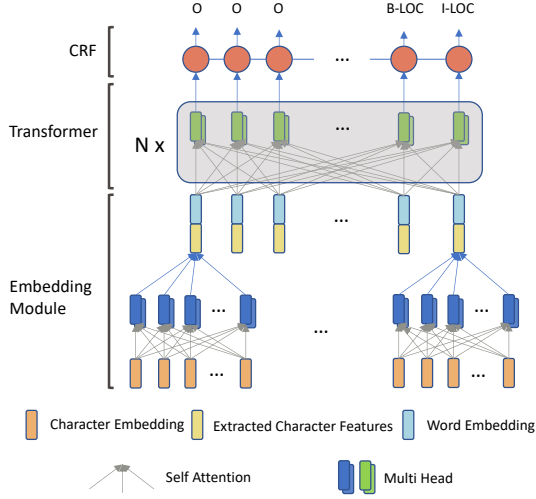
Figure 2: Model structure of TENER for English NER tasks. In TENER, Transformer encoder is used not only to extract the word-level contextual information, but also to encode character-level information in a word.

3 Proposed Model

In this paper, we utilize the Transformer encoder to model the long-range and complicated interactions of sentence for NER. The structure of proposed model is shown in Fig 2. We detail each parts in the following sections.

3.1 Embedding Layer

To alleviate the problems of data sparsity and out-of-vocabulary (OOV), most NER models adopted the CNN character encoder (Ma and Hovy, 2016; Ye and Ling, 2018; Chen et al., 2019) to represent words. Compared to BiLSTM based character encoder (Lample et al., 2016; Ghaddar and Langlais, 2018), CNN is more efficient. Since Transformer can also fully exploit the GPU's parallelism, it is interesting to use Transformer as the character encoder. A potential benefit of Transformer-based character encoder is to extract different n-grams and even uncontinuous character patterns, like "un..ily" in "unhappily" and "uneasily". For the model's uniformity, we use the "adapted Transformer" to represent the Transformer introduced in next subsection.

The final word embedding is the concatenation of the character features extracted by the character encoder and the pre-trained word embeddings.

3.2 Encoding Layer with Adapted Transformer

Although Transformer encoder has potential advantage in modeling long-range context, it is not working well for NER task. In this paper, we propose an adapted Transformer for NER task with two improvements.

3.2.1 Direction- and Distance-Aware Attention

Inspired by the success of BiLSTM in NER tasks, we consider what properties the Transformer lacks compared to BiLSTM-based models. One observation is that BiLSTM can discriminatively collect the context information of a token from its left and right sides. But it is not easy for the Transformer to distinguish which side the context information comes from.

Although the dot product between two sinusoidal position embeddings is able to reflect their distance, it lacks directionality and this property will be broken by the vanilla Transformer attention. To illustrate this, we first prove two properties of the sinusoidal position embeddings.

Property 1. *For an offset k and a position t, $PE_{t+k}^T PE_t$ only depends on k, which means the dot product of two sinusoidal position embeddings can reflect the distance between two tokens.*

Proof. Based on the definitions of Eq.(8) and Eq.(9), the position embedding of t-th token is

$$PE_t = \begin{bmatrix} \sin(c_0 t) \\ \cos(c_0 t) \\ \vdots \\ \sin(c_{\frac{d}{2}-1} t) \\ \cos(c_{\frac{d}{2}-1} t) \end{bmatrix}, \qquad (10)$$

where d is the dimension of the position embedding, c_i is a constant decided by i, and its value is $1/10000^{2i/d}$.

Therefore,

$$PE_t^T PE_{t+k} = \sum_{j=0}^{\frac{d}{2}-1} [\sin(c_j t) \sin(c_j (t+k)) \\ + \cos(c_j t) \cos(c_j (t+k))] \qquad (11)$$

$$= \sum_{j=0}^{\frac{d}{2}-1} \cos(c_j (t - (t+k))) \qquad (12)$$

$$= \sum_{j=0}^{\frac{d}{2}-1} \cos(c_j k), \qquad (13)$$

where Eq.(11) to Eq.(12) is based on the equation $\cos(x - y) = \sin(x)\sin(y) + \cos(x)\cos(y)$. $\square$
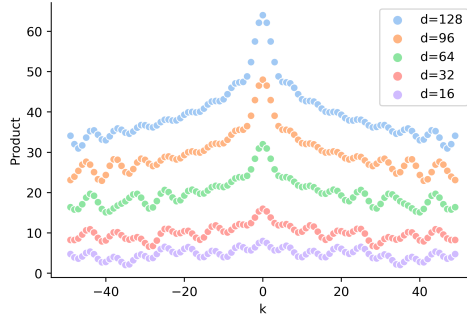
Figure 3: Dot product between two sinusoidal position embeddings whose distance is k. It is clear that the product is symmetrical, and with the increment of $|k|$, it has a trend to decrease, but this decrease is not monotonous.
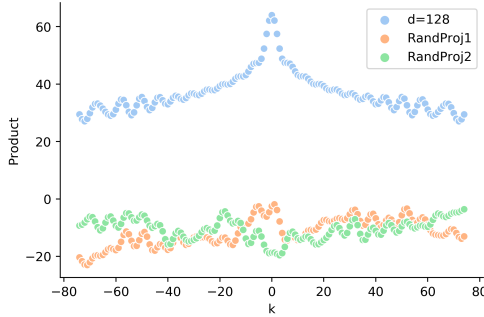

Figure 4: The upper line is the product between $PE_t^T PE_{t+k}$. The lower two lines are the products of $PE_t^T W PE_{t+k}$ with two random Ws. Although $PE_t^T PE_{t+k}$ can reflect the distance, the $PE_t^T W PE_{t+k}$ has no clear pattern.

Property 2. *For an offset k and a position t, $PE_t^T PE_{t-k} = PE_t^T PE_{t+k}$, which means the sinusoidal position embeddings is unware of directionality.*

Proof. Let $j = t - k$, according to property 1, we have

$$PE_t^T PE_{t+k} = PE_j^T PE_{j+k} \tag{14}$$
$$= PE_{t-k}^T PE_t. \tag{15}$$

$\square$

The relation between d, k and $PE_t^T PE_{t+k}$ is displayed in Fig 3. The sinusoidal position embeddings are distance-aware but lacks directionality.

However, the property of distance-awareness also disappears when PE_t is projected into the query and key space of self-attention. Since in vanilla Transformer the calculation between PE_t and PE_{t+k} is actually $PE_t^T W_q^T W_k PE_{t+k}$, where

W_q, W_k are parameters in Eq.(1). Mathematically, it can be viewed as $PE_t^T W PE_{t+k}$ with only one parameter W. The relation between $PE_t^T PE_{t+k}$ and $PE_t^T W PE_{t+k}$ is depicted in Fig 4.

Therefore, to improve the Transformer with direction- and distance-aware characteristic, we calculate the attention scores using the equations below:

$$Q, K, V = HW_q, H_{d_k}, HW_v, \tag{16}$$
$$R_{t-j} = [\dots \sin(\frac{t-j}{10000^{2i/d_k}}) \cos(\frac{t-j}{10000^{2i/d_k}}) \dots]^T, \tag{17}$$
$$A_{t,j}^{rel} = Q_t K_j^T + Q_t R_{t-j}^T + \mathbf{u}K_j^T + \mathbf{v}R_{t-j}^T, \tag{18}$$
$$\text{Attn}(Q,K,V) = \text{softmax}(A^{rel})V, \tag{19}$$

where t is index of the target token, j is the index of the context token, Q_t, K_j is the query vector and key vector of token t, j respectively, $W_q, W_v \in \mathbb{R}^{d \times d_k}$. To get $H_{d_k} \in \mathbb{R}^{l \times d_k}$, we first split H into d/d_k partitions in the second dimension, then for each head we use one partition. $\mathbf{u} \in \mathbb{R}^{d_k}$, $\mathbf{v} \in \mathbb{R}^{d_k}$ are learnable parameters, R_{t-j} is the relative positional encoding, and $R_{t-j} \in \mathbb{R}^{d_k}$, i in Eq.(17) is in the range $[0, \frac{d_k}{2}]$. $Q_t^T K_j$ in Eq.(18) is the attention score between two tokens; $Q_t^T R_{t-j}$ is the tth token's bias on certain relative distance; $u^T K_j$ is the bias on the jth token; $v^T R_{t-j}$ is the bias term for certain distance and direction.

Based on Eq.(17), we have

$$R_t, R_{-t} = \begin{bmatrix} \sin(c_0 t) \\ \cos(c_0 t) \\ \vdots \\ \sin(c_{\frac{d}{2}-1} t) \\ \cos(c_{\frac{d}{2}-1} t) \end{bmatrix}, \begin{bmatrix} -\sin(c_0 t) \\ \cos(c_0 t) \\ \vdots \\ -\sin(c_{\frac{d}{2}-1} t) \\ \cos(c_{\frac{d}{2}-1} t) \end{bmatrix}, \tag{20}$$

because $\sin(-x) = -\sin(x), \cos(x) = \cos(-x)$. This means for an offset t, the forward and backward relative positional encoding are the same with respect to the $\cos(c_i t)$ terms, but is the opposite with respect to the $\sin(c_i t)$ terms. Therefore, by using R_{t-j}, the attention score can distinguish different directions and distances.

The above improvement is based on the work (Shaw et al., 2018; Dai et al., 2019). Since the size of NER datasets is usually small, we avoid direct multiplication of two learnable parameters, because they can be represented by one learnable parameter. Therefore we do not use W_k in Eq.(16). The multi-head version is the same as Eq.(6), but we discard W_o since it is directly multiplied by W_1 in Eq.(7).

Table 1: Details of Datasets.

	Dataset	Type	Train	Dev	Test
English	CoNLL2003	Sentence	14.0k	3.2k	3.5k
		Token	203.6k	51.4k	46.4k
	OntoNotes 5.0	Sentence	59.9k	8.5k	8.3k
		Token	1088.5k	147.7k	152.7k
Chinese	OntoNotes 4.0	Sentence	15.7k	4.3k	4.3k
		Token	491.9k	200.5k	208.1k
	MSRA	Sentence	46.4k	4.4k	4.4k
		Token	2169.9k	172.6k	172.6k
	Weibo	Sentence	1.4k	0.3k	0.3k
		Token	73.5k	14.4k	14.8k
	Resume	Sentence	3.8k	0.5k	0.5k
		Token	124.1k	13.9k	15.1k

3.2.2 Un-scaled Dot-Product Attention

The vanilla Transformer use the scaled dot-product attention to smooth the output of softmax function. In Eq.(3), the dot product of key and value matrices is divided by the scaling factor $\sqrt{d_k}$.

We empirically found that models perform better without the scaling factor $\sqrt{d_k}$. We presume this is because without the scaling factor the attention will be sharper. And the sharper attention might be beneficial in the NER task since only few words in the sentence are named entities.

3.3 CRF Layer

In order to take advantage of dependency between different tags, the Conditional Random Field (CRF) was used in all of our models. Given a sequence $\mathbf{s} = [s_1, s_2, ..., s_T]$, the corresponding golden label sequence is $\mathbf{y} = [y_1, y_2, ..., y_T]$, and $\mathbf{Y}(\mathbf{s})$ represents all valid label sequences. The probability of $\mathbf{y}$ is calculated by the following equation

$$P(\mathbf{y}|\mathbf{s}) = \frac{\sum_{t=1}^{T} e^{f(\mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{s})}}{\sum_{\mathbf{y}'}^{\mathbf{Y}(\mathbf{s})} \sum_{t=1}^{T} e^{f(\mathbf{y}'_{t-1}, \mathbf{y}'_t, \mathbf{s})}}, \qquad (21)$$

where $f(\mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{s})$ computes the transition score from $\mathbf{y}_{t-1}$ to $\mathbf{y}_t$ and the score for $\mathbf{y}_t$. The optimization target is to maximize $P(\mathbf{y}|\mathbf{s})$. When decoding, the Viterbi Algorithm is used to find the path achieves the maximum probability.

4 Experiment

4.1 Data

We evaluate our model in two English NER datasets and four Chinese NER datasets.

(1) CoNLL2003 is one of the most evaluated English NER datasets, which contains four different named entities: PERSON, LOCATION, ORGANIZATION, and MISC (Sang and Meulder, 2003).

(2) OntoNotes 5.0 is an English NER dataset whose corpus comes from different domains, such as telephone conversation, newswire. We exclude the New Testaments portion since there is no named entity in it (Chen et al., 2019; Chiu and Nichols, 2016). This dataset has eleven entity names and seven value types, like CARDINAL, MONEY, LOC.

(3) Weischedel (2011) released OntoNotes 4.0. In this paper, we use the Chinese part. We adopted the same pre-process as (Che et al., 2013).

(4) The corpus of the Chinese NER dataset MSRA came from news domain (Levow, 2006).

(5) Weibo NER was built based on text in Chinese social media Sina Weibo (Peng and Dredze, 2015), and it contained 4 kinds of entities.

(6) Resume NER was annotated by (Zhang and Yang, 2018).

Their statistics are listed in Table 1. For all datasets, we replace all digits with "0", and use the BIOES tag schema. For English, we use the Glove 100d pre-trained embedding (Pennington et al., 2014). For the character encoder, we use 30d randomly initialized character embeddings. More details on models' hyper-parameters can be found in the supplementary material. For Chinese, we used the character embedding and bigram embedding released by (Zhang and Yang, 2018). All pretrained embeddings are finetuned during training. In order to reduce the impact of randomness, we ran all of our experiments at least three times, and its average F1 score and standard deviation are reported.

We used random-search to find the optimal hyper-parameters, hyper-parameters and their ranges are displayed in the supplemental material. We use SGD and 0.9 momentum to optimize the model. We run 100 epochs and each batch has 16 samples. During the optimization, we use the triangle learning rate (Smith, 2017) where the learning rate rises to the pre-set learning rate at the first 1% steps and decreases to 0 in the left 99% steps. The model achieves the highest development performance was used to evaluate the test set. The hyper-parameter search range and other settings can be found in the supplementary material. Codes are available at `https://github.com/fastnlp/TENER`.

Models	Weibo	Resume	OntoNotes4.0	MSRA
BiLSTM ♣	56.75	94.41	71.81	91.87
ID-CNN ♠	-	93.75	62.25	-
CAN-NER* (Zhu and Wang, 2019)	59.31	94.94	73.64	92.97
Transformer	46.38 ± 0.78	93.43 ± 0.26	66.49 ± 0.30	88.35 ± 0.60
TENER(Ours)	$\mathbf{58.17 \pm 0.22}$	$\mathbf{95.00 \pm 0.25}$	$\mathbf{72.43 \pm 0.39}$	$\mathbf{92.74 \pm 0.27}$
w/ scale	57.40 ± 0.3	94.00 ± 0.51	71.72 ± 0.08	91.67 ± 0.23

Table 2: The F1 scores on Chinese NER datasets. ♣,♠ are results reported in (Zhang and Yang, 2018) and (Gui et al., 2019a), respectively. "w/ scale" means TENER using the scaled attention in Eq.(19). * their results are not directly comparable with ours, since they used 100d pre-trained character and bigram embeddings. Other models use the same embeddings.

4.2 Results on Chinese NER Datasets

We first present our results in the four Chinese NER datasets. Since Chinese NER is directly based on the characters, it is more straightforward to show the abilities of different models without considering the influence of word representation.

As shown in Table 2, the vanilla Transformer does not perform well and is worse than the BiL-STM and CNN based models. However, when relative positional encoding combined, the performance was enhanced greatly, resulting in better results than the BiLSTM and CNN in all datasets. The number of training examples of the Weibo dataset is tiny, therefore the performance of the Transformer is abysmal, which is as expected since the Transformer is data-hungry. Nevertheless, when enhanced with the relative positional encoding and unscaled attention, it can achieve even better performance than the BiLSTM-based model. The superior performance of the adapted Transformer in four datasets ranging from small datasets to big datasets depicts that the adapted Transformer is more robust to the number of training examples than the vanilla Transformer. As the last line of Table 2 depicts, the scaled attention will deteriorate the performance.

4.3 Results on English NER datasets

The comparison between different NER models on English NER datasets is shown in Table 3. The poor performance of the Transformer in the NER datasets was also reported by (Guo et al., 2019). Although performance of the Transformer is higher than (Guo et al., 2019), it still lags behind the BiLSTM-based models (Ma and Hovy, 2016). Nonetheless, the performance is massively enhanced by incorporating the relative positional encoding and unscaled attention into the Transformer. The adaptation not only makes the Transformer achieve superior performance than BiL-

Models	CoNLL2003	OntoNotes 5.0
BiLSTM-CRF (Huang et al., 2015)	88.83	
CNN-BiLSTM-CRF (Chiu and Nichols, 2016)	90.91 ± 0.20	86.12 ± 0.22
BiLSTM-BiLSTM-CRF (Lample et al., 2016)	90.94	
CNN-BiLSTM-CRF (Ma and Hovy, 2016)	91.21	
ID-CNN (Strubell et al., 2017)	90.54 ± 0.18	86.84 ± 0.19
LM-LSTM-CRF (Liu et al., 2018)	91.24 ± 0.12	
CRF+HSCRF (Ye and Ling, 2018)	91.26 ± 0.1	
BiLSTM-BiLSTM-CRF (Akhundov et al., 2018)	91.11	
LS+BiLSTM-CRF (Ghaddar and Langlais, 2018)	90.52 ± 0.20	86.57 ± 0.1
CN^3 (Liu et al., 2019)	91.1	
GRN (Chen et al., 2019)	91.44 ± 0.16	87.67 ± 0.17
Transformer	89.57 ± 0.12	86.73 ± 0.07
TENER (Ours)	91.33 ± 0.05	$\mathbf{88.43 \pm 0.12}$
w/ scale	91.06 ± 0.09	87.94 ± 0.1
w/ CNN-char	$\mathbf{91.45 \pm 0.07}$	88.25 ± 0.11

Table 3: The F1 scores on English NER datasets. We only list results based on non-contextualized embeddings, and methods utilized pre-trained language models, pre-trained features, or higher dimension word vectors are excluded. TENER (Ours) uses the Transformer encoder both in the character-level and word-level. "w/ scale" means TENER using the scaled attention in Eq.(19). "w/ CNN-char" means TENER using CNN as character encoder instead of AdaTrans.

STM based models, but also unveil the new state-of-the-art performance in two NER datasets when only the Glove 100d embedding and CNN character embedding are used. The same deterioration of performance was observed when using the scaled attention. Besides, if ELMo was used (Peters et al., 2018), the performance of TENER can be further boosted as depicted in Table 4.

Models	CoNLL2003	OntoNotes 5.0
BiLSTM	92.55±0.10	88.88±0.16
GRN (Chen et al., 2019)	92.34±0.1	-
TENER (Ours)	**92.62±0.09**	**89.78±0.15**

Table 4: Performance of models with ELMo as their embeddings in English NER datasets. "BiLSTM" is our run. In the larger OntoNotes5.0, TENER achieves much better F1 score.

Char \ Word	BiLSTM	ID-CNN	AdaTrans
No Char	88.34 ± 0.32	87.30 ± 0.15	88.37 ± 0.27
BiLSTM	91.32 ± 0.13	89.99 ± 0.14	91.29 ± 0.12
CNN	91.22 ± 0.10	90.17 ± 0.02	$\mathbf{91.45 \pm 0.07}$
Transformer	91.12 ± 0.10	90.05 ± 0.13	91.23 ± 0.06
AdaTrans	91.38 ± 0.15	89.99 ± 0.05	91.33 ± 0.05

(a) CoNLL2003

Char \ Word	BiLSTM	ID-CNN	AdaTrans
No Char	85.20 ± 0.23	84.26 ± 0.07	85.80 ± 0.10
BiLSTM	87.85 ± 0.09	87.38 ± 0.17	88.12 ± 0.16
CNN	87.79 ± 0.14	87.10 ± 0.06	88.25 ± 0.11
Transformer	88.01 ± 0.06	87.31 ± 0.10	88.20 ± 0.07
AdaTrans	88.12 ± 0.17	87.51 ± 0.11	$\mathbf{88.43 \pm 0.12}$

(b) OntoNotes 5.0

Table 5: F1 scores in the CoNLL2003 and OntoNotes 5.0. "Char" means character-level encoder, and "Word" means word-level encoder. "AdaTrans" means our adapted Transformer encoder.

4.4 Analysis of Different Character Encoders

The character-level encoder has been widely used in the English NER task to alleviate the data sparsity and OOV problem in word representation. In this section, we cross different character-level encoders (BiLSTM, CNN, Transformer encoder and our adapted Transformer encoder (AdaTrans for short)) and different word-level encoders (BiLSTM, ID-CNN and AdaTrans) to implement the NER task. Results on CoNLL2003 and OntoNotes 5.0 are presented in Table 5a and Table 5b, respectively.

The ID-CNN encoder is from (Strubell et al., 2017), and we re-implement their model in Py-Torch. For different combinations, we use random search to find its best hyper-parameters. Hyperparameters for character encoders were fixed. The details can be found in the supplementary material.

For the results on CoNLL2003 dataset which is depicted in Table 5a, the AdaTrans performs as good as the BiLSTM in different character en-

coder scenario averagely. In addition, from Table 5b, we can find the pattern that the AdaTrans character encoder outpaces the BiLSTM and CNN character encoders when different word-level encoders being used. Moreover, no matter what character encoder being used or none being used, the AdaTrans word-level encoder gets the best performance. This implies that when the number of training examples increases, the AdaTrans character-level and word-level encoder can better realize their ability.

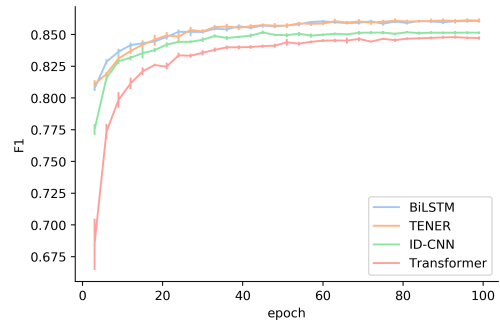### 4.5 Convergent Speed Comparison

Figure 5: Convergent speed in the development dataset of OntoNotes 5.0 for four kinds of models.

We compare the convergent speed of BiLSTM, ID-CNN, Transformer, and TENER in the development set of the OntoNotes 5.0. The curves are shown in Fig 5. TENER converges as fast as the BiLSTM model and outperforms the vanilla Transformer.

5 Conclusion

In this paper, we propose TENER, a model adopting Transformer Encoder with specific customizations for the NER task. Transformer Encoder has a powerful ability to capture the long-range context. In order to make the Transformer more suitable to the NER task, we introduce the direction-aware, distance-aware and un-scaled attention. Experiments in two English NER tasks and four Chinese NER tasks show that the performance can be massively increased. Under the same pretrained embeddings and external knowledge, our proposed modification outperforms previous models in the six datasets. Meanwhile, we also found the adapted Transformer is suitable for being used as the English character encoder, because it has the potentiality to extract intricate patterns from char-

acters. Experiments in two English NER datasets show that the adapted Transformer character encoder performs better than BiLSTM and CNN character encoders.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING*, pages 1638–1649.

Adnan Akhundov, Dietrich Trautmann, and Georg Groh. 2018. Sequence labeling: A practical approach. *CoRR*, abs/1808.03926.

Wanxiang Che, Mengqiu Wang, Christopher D. Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *NAACL*, pages 52–62.

Hui Chen, Zijia Lin, Guiguang Ding, Jianguang Lou, Yusen Zhang, and Borje Karlsson. 2019. GRN: Gated relation network to enhance convolutional neural network for named entity recognition. In *AAAI*.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*, 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(Aug):2493–2537.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based LSTM-CRF with radical-level features for chinese named entity recognition. In *NLPCC*, pages 239–250.

Pavlina Fragkou. 2017. Applying named entity recognition and co-reference resolution for segmenting english texts. *Progress in Artificial Intelligence*, 6(4):325–346.

Abbas Ghaddar and Philippe Langlais. 2018. Robust lexical features for improved neural network named-entity recognition. *arXiv preprint arXiv:1806.03489*.

Tao Gui, Ruotian Ma, Qi Zhang, Lujun Zhao, Yu-Gang Jiang, and Xuanjing Huang. 2019a. Cnn-based chinese NER with lexicon rethinking. In *IJCAI*, pages 4982–4988.

Tao Gui, Yicheng Zou, Qi Zhang, Minlong Peng, Jinlan Fu, Zhongyu Wei, and Xuanjing Huang. 2019b. A lexicon-based graph neural network for chinese ner.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. In *NAACL*, pages 1315–1325.

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

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2019. Music transformer: Generating music with long-term structure. In *ICLR*.

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

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth Workshop on Chinese Language Processing, SIGHAN@COLING/ACL 2006, Sydney, Australia, July 22-23, 2006*, pages 108–117.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *AAAI*, pages 5253–5260.

Pengfei Liu, Shuaichen Chang, Xuanjing Huang, Jian Tang, and Jackie Chi Kit Cheung. 2019. Contextualized non-local neural networks for sequence learning. In *AAAI*, volume 33, pages 6762–6769.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*, volume 1, pages 1064–1074.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *EMNLP*, pages 548–554.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NACCL*, volume 1, pages 2227–2237.

Matthew E. Peters, Waleed Ammar, Chandra Bhaga-vatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL (1)*, pages 1756–1765. ACL.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *NAACL*, pages 142–147.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *NAACL*, pages 464–468.

Leslie N. Smith. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, Santa Rosa, CA, USA, March 24-31, 2017*, pages 464–472.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. *arXiv preprint arXiv:1702.02098*.

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.

Ralph Weischedel. 2011. Ontonotes release 4.0 LDC2011T03.

Yingwei Xin, Ethan Hart, Vibhuti Mahajan, and Jean-David Ruvini. 2018. Learning better internal structure of words for sequence labeling. In *EMNLP*, pages 2584–2593.

Fan Yang, Jianhu Zhang, Gongshen Liu, Jie Zhou, Cheng Zhou, and Huanrong Sun. 2018. Five-stroke based cnn-birnn-crf network for chinese named entity recognition. In *NLPCC*, pages 184–195.

	English
number of layers	[1, 2]
hidden size	[200, 400, 600, 800, 1200]
learning rate	[0.01, 0.007, 0.005]
fc dropout	0.4

Table 6: The hyper-parameters and hyper-parameter search ranges for BiLSTM.

Zhi-Xiu Ye and Zhen-Hua Ling. 2018. Hybrid semi-markov CRF for neural sequence labeling. *arXiv preprint arXiv:1805.03838*.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state LSTM for text representation. In *ACL (1)*, pages 317–327. Association for Computational Linguistics.

Yue Zhang and Jie Yang. 2018. Chinese NER using lattice LSTM. In *ACL*, pages 1554–1564.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *NLPCC*, pages 662–671.

Yuying Zhu and Guoxin Wang. 2019. CAN-NER: convolutional attention network for chinese named entity recognition. In *NAACL*, pages 3384–3393.

6 Supplemental Material

6.1 Character Encoder

We exploit four kinds of character encoders. For all character encoders, the randomly initialized character embeddings are 30d. The hidden size of BiLSTM used in the character encoder is 50d in each direction. The kernel size of CNN used in the character encoder is 3, and we used 30 kernels with stride 1. For Transformer and adapted Transformer, the number of heads is 3, and every head is 10d, the dropout rate is 0.15, the feed-forward dimension is 60. The Transformer used the sinusoid position embedding. The number of parameters for the character encoder (excluding character embedding) when using BiLSTM, CNN, Transformer and adapted Transformer are 35830, 3660, 8460 and 6600 respectively. For all experiments, the hyper-parameters of character encoders stay unchanged.

6.2 Hyper-parameters

The hyper-parameters and search ranges for different encoders are presented in Table 6, Table 7 and Table 8.

	English
number of layers	[2, 3, 4, 5, 6]
number of kernels	[200, 400, 600, 800]
kernel size	3
learning rate	[2e-3, 1.5e-3, 1e-3, 7e-4]
fc dropout	0.4

Table 7: The hyper-parameters and hyper-parameter search ranges for ID-CNN.

	Chinese	English
number of layers	[1, 2]	[1, 2]
number of head	[4, 6, 8, 10]	[8, 10, 12, 14]
head dimension	[32, 48, 64, 80, 96]	[64, 80, 96, 112, 128]
learning rate	[1e-3, 5e-4, 7e-4]	[9e-4, 7e-4, 5e-4]
transformer dropout	0.15	0.15
fc dropout	0.4	0.4

Table 8: The hyper-parameters and hyper-parameter search ranges for Transformer and adapted Transformer in Chinese and English NER datasets.