CLGP: Multi-Feature Embedding based Cross-Attention for Chinese NER

Anonymous ACL submission

Abstract

The previous works fused lexicon information while ignored two important Chinese language characteristics: glyph and pinyin, which carry significant syntax and semantics information for sequence tagging tasks. This paper proposes *CLGP*[1], which utilizes four specific extractors to obtain the embeddings of character, glyph, pinyin and lexicon, and further uses a network based on cross-attention to perform multi-embedding fusion. Specifically, we introduce the embedding scheme to preserve the lexicon matching results, and design two specific CNN to extract glyph and pinyin embeddings. Moreover, we fuse the four embeddings by cross-attention based network to enhance the Chinese NER. The experimental results on four famous datasets show that *CLGP* achieves the state-of-the-art (SOTA) performance.

1 Introduction

Named Entity Recognition(NER) aims to automatically recognize named entities in plain text. In English NER, LSTM-CRF models (Lample et al., 2016; Ma and Hovy, 2016; Liu et al., 2018; Chiu and Nichols, 2016; Huang et al., 2015) have achieved the state-of-the-art results by integrating character information into word representations.

Compared with English NER, Chinese NER is more complicated because it has no obvious word boundaries. One intuitive way is first to perform word segmentation using existing Chinese word segmentation tools and then apply the word-based NER model (Jie et al., 2016; He and Sun, 2017b). However, such methods suffer from error propagation because named entities may encounter out-of-vocabulary problems in segmentation. Consequently, some works show that the character-based methods for Chinese NER have been empirically proven to be effective (He and Wang, 2008; Liu et al., 2010; Li et al., 2014; Sui et al., 2019; Ding et al., 2019; Liu et al., 2019). A drawback of the character-based NER model is that explicit word information is not fully exploited. To solve this issue, Zhang and Yang (2018) proposed Lattice-LSTM that incorporates the word information into the character-based NER model to avoid the error propagation of word segmentation.

However, Lattice-LSTM still faces several challenges. First, it is limited by the structure of non-parallelizable sequential LSTM. Second, the model architecture is not only quite complicated but also difficult to transfer to other neural network architectures. With this consideration, the LR-CNN model (Gui et al., 2019a) fully utilizes the parallel computation of GPU. Ma et al. (2020) proposed a more straightforward method to implement the idea of Lattice-LSTM. Moreover, FLAT (Li et al., 2020) converts the lattice structure into a flat structure that enables characters to interact with any potential words directly. However, the NER task is initially designed for English without considering the language characteristics of Chinese. Therefore, two critical aspects specific to the Chinese language are missing in the current lexicon-based Chinese NER model: glyph and pinyin information. Previous works (Sun et al., 2014; Shi et al., 2015; Liu et al., 2017; Dai and Cai, 2017; Su and Lee, 2017; Meng et al., 2019; Sehanobish and Song, 2020; Xuan et al., 2020; Wu et al., 2021b; Sun et al., 2021) proved that the rich semantics behind Chinese character glyph can enrich the expressiveness of the Chinese NER task. As shown in Table 1,“湖(lake)”, “海(sea)” and “湾(bay)” all have the radical “氵(water)”, which indicates that they are all related to water in semantics. Furthermore, other works (Meng et al., 2019; Sun et al., 2021) showed that pinyin is very important in modeling semantics features, which cannot be captured by contextualized or glyph. As can be seen in Table 2, the Chinese character “行” has two distinctly

[1]The source code of the proposed method is publicly available at: https://github.com/acl-2022/CLGP

Rds	Characters	Meaning
氵	湖(lake),海(sea),湾(bay)	水(water)
⺾	茶(tea),芽(bud),荔(litchi)	草(grass)
鸟	鹏(roc),鸦(crow),鸭(duck)	鸟(birds)
口	吃(eat),喊(shout),吐(spit)	口(mouth)

Table 1: Some examples of Chinese radicals, including "氵"(water), "草"(grass),"鸟"(birds),"口"(mouth).

Character	Pronunciation	Meaning
行	xíng, háng	walking, line
乐	yuè, lè	music, happy
重	zhòng, chóng	weight, repeat
好	hào, hǎo	like, good

Table 2: In different context semantics, different pronunciations represent different meanings, such as "行","乐","重","好".

different pronunciations: One can be pronounced as "xíng" , which means "walking"; and the other is "háng" , which means "line".

This paper proposes a novel multi-feature fusion model. We use the embedding scheme to preserve the lexicon matching results, and design two specific CNN architectures to extract glyph and pinyin embeddings. Moreover, we deeply fuse the four embeddings by cross-attention based network to enrich the expressiveness of the Chinese NER. The main contributions of this work are as follows:

- The use of multi-feature embedding of the Chinese characters enhances NER task.
- We proposed *CLGP*, a novel multi-feature embedding fusion model, which first extracts different features from character, lexicon, glyph, and pinyin information, then deeply fuses them by cross-attention based network.
- *CLGP* has flexible scalability that can be widely used in various sequence architectures, and can be easily combined with pre-trained models (such as BERT).
- The experimental results show that our *CLGP* yields SOTA results on four well-known Chinese NER benchmark datasets.

2 Related Work

There are two main types of Chinese NER enhancement methods, including word segmentation information fusion and glyph feature fusion.

2.1 Lexicon-based Chinese NER

Zhang and Yang (2018)[2] proposed a Lattice-LSTM that integrates latent word information into character-based LSTM-CRF. Gui et al. (2019a) introduced a CNN with a rethinking mechanism to model all the characters and potential words that matched the sentence in parallel. However, those methods are limited by efficiency and cannot be fixed long-distance dependencies issues. As a result, other methods converted the lattice structure into the graph and used a graph neural network (GNN) to encode it, such as Lexicon-based Graph Network (LGN) (Gui et al., 2019b) and Collaborative Graph Network (CGN) (Sui et al., 2019). However, the above methods need use LSTM as the bottom encoder, which makes the model complicated. Based on this consideration, SoftLexicon (Ma et al., 2020)[3] implemented the Lattice-LSTM by a simpler approach that incorporates all the matched words for each character to a character-based NER model in order to avoid complicated model architecture. Moreover, FLAT (Li et al., 2020) also simplified the Lattice architecture that designs an ingenious position encoding for the lattice-structure to reconstruct a lattice from a set of tokens. Nevertheless, unlike the English language, Chinese has its syntax, lexicon, and pronunciation characteristics. Therefore, the Chinese language features (glyph and pinyin) should be considered to enrich the expressiveness of the Chinese NER task.

2.2 Glyph-based Chinese NER

As a logographic language, the Chinese characters encode rich information of their meanings. Intuitively, the rich semantics behind Chinese character glyphs can enhance the expressiveness of NLP tasks. In order to improve the model's performance in Chinese NLP taks, some works (Mikolov et al., 2013; Sun et al., 2014; Shi et al., 2015; Li et al., 2015; Yin et al., 2016; Dong et al., 2016) used indexed radical embedding to capture Chinese character semantics. However, other works (Liu et al., 2017; Shao et al., 2017; Zhang and LeCun, 2017; Dai and Cai, 2017) utilized CNNs to extract glyph features from character images. Besides, Meng et al. (2019) proposed glyph-vectors for Chinese character representations. This vector employs several historical Chinese scripts to enrich the pic-

[2] https://github.com/jiesutd/LatticeLSTM.

[3] https://github.com/v-mipeng/LexiconAugmentedNER.

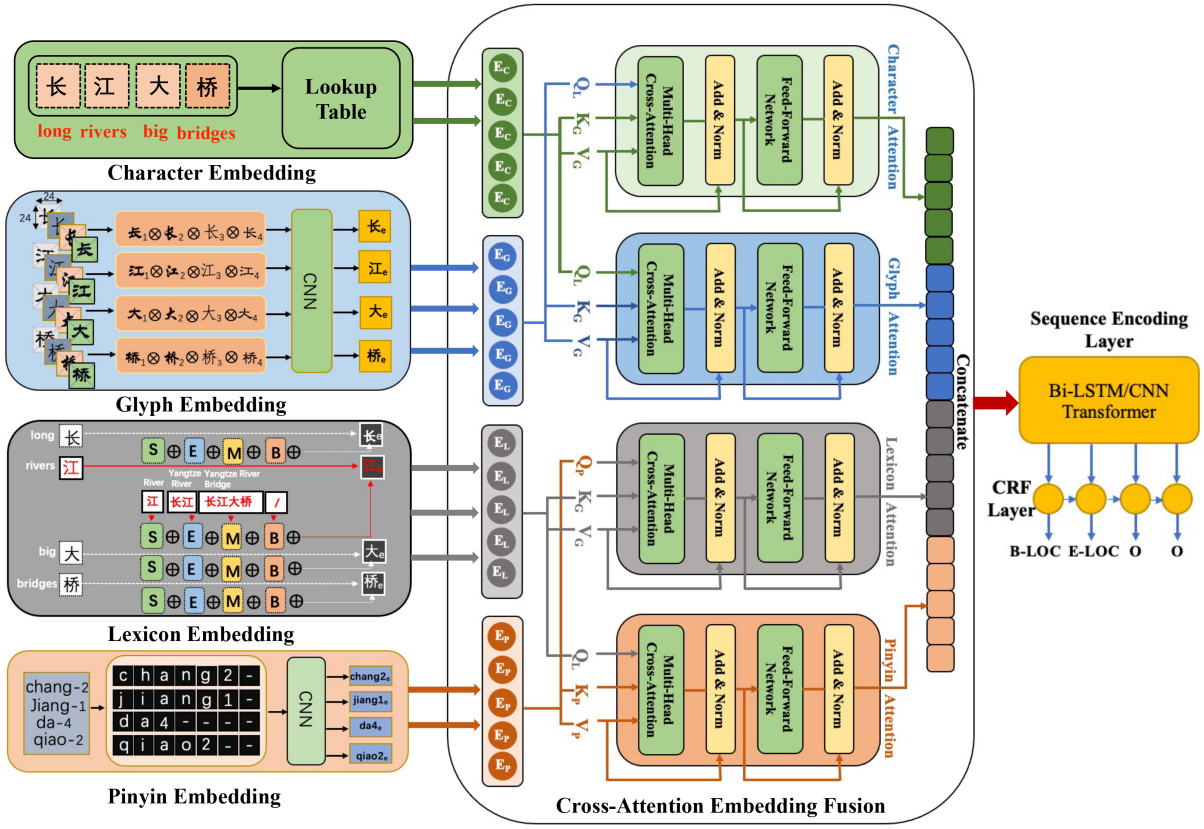

Figure 1: The overall architecture of CLGP, where C, L, G and P are the initial letters of Character, Lexicon, Glyph and Pinyin, respectively. Q, K and V are the query vector, key vector and value vector in the attention mechanism.

tographic evidence in character. Inspired by the idea of (Meng et al., 2019; Xuan et al., 2020; Sehanobish and Song, 2020; Sun et al., 2021; Wu et al., 2021a), we find that fusing the glyph into the Chinese NER can achieve good performance. However, it is hard to yield SOTA results that only exploit the glyph features.

The feature extraction method and information fusion mode have a great impact on the performance. Therefore, we propose the embedding scheme to preserve the lexicon matching results, and design two specific CNN architectures to extract glyph and pinyin embeddings. We further fuse the aforementioned embeddings by cross-attention based network to enrich the Chinese NER.

3 Model

In this section, we mainly introduce *CLGP* in detail. The overall architecture is shown in Figure 1, which consists of four parts. The first part is a feature extractor that generates four different embeddings, the second part is cross-attention based fusion network, the rest are the sequence modeling and decoding layer.

3.1 Input Embedding layer

The input presentation layer is an important part of our work, including the embedding generation models of Character, Lexicon, Glyph and Pinyin.

3.1.1 Character Embedding

For character based Chinese NER model, input sequence can be viewed as a character set $s=\{c_1,c_2,...,c_n\} \in D_c$, where D_c denotes the character vocabulary. Each character c_i can be embedded as follow:

$$x_i^c = e^c(c_i). \quad (1)$$

where e^c represents the character embedding lookup table.

3.1.2 Lexicon Embedding

The lexicon-based embedding is constructed in two steps. First, we classify all matched words of each character into BMES sets. Then, we fuse each word sets into a N-dimensional vector and add them into corresponding character representation.

Classification matched words. For each character c_i in the input sentence $s=\{c_1,c_2,...,c_n\}$, the

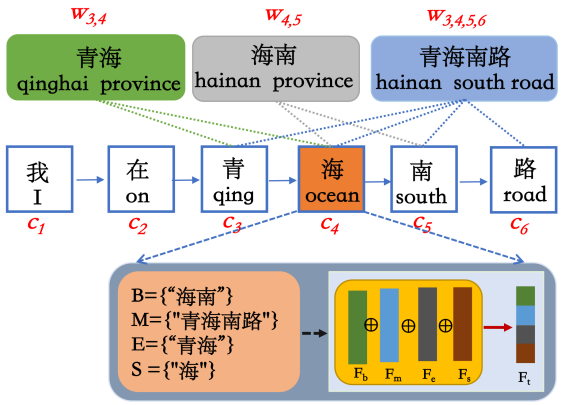

Figure 2: The Lexicon Embedding. F_b, F_m, and F_s denote the vectors obtained by adding all word vectors in the corresponding group of the BMES set, and F_s is the embedding of this character. Besides, $\oplus$ represents the point-wise addition, and F_t is the final fusion vector.

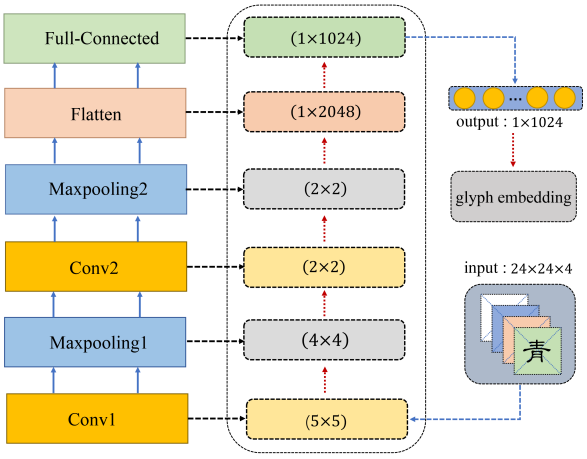

Figure 3: CNN Architecture for Glyph Embedding.

BMES set can be built as follow:

$$B(c_i) \leftarrow \{\forall w_j \in L, w_j \Rightarrow [c_i, c_m, c_e]\}, \quad (2)$$

$$M(c_i) \leftarrow \{\forall w_j \in L, w_j \Rightarrow [c_b, c_i, c_e]\}, \quad (3)$$

$$E(c_i) \leftarrow \{\forall w_j \in L, w_j \Rightarrow [c_b, c_m, c_i]\}, \quad (4)$$

$$S(c_i) \leftarrow \{\forall c_i \in L\}. \quad (5)$$

where L represents the lexicon that we apply in this work, $w_j \Rightarrow [c_i,c_m,c_e]$ denotes c_i in the starting position of w_j, c_m and c_e are the characters of middle and end positions in w_j, and $S(c_i)$ stands for c_i itself. Especially, if a word set is empty, it is represented by symbol '/'.

Fusion all word sets. After classifying the "BMES" word set of each character, we map them into a N-dimensional vector. In order to simplify the structure as much as possible, we use point-wise addition to combine the word set features, which can not only simply implement but also effectively improve computational efficiency compared with weight-based methods. As an example shown in Figure 2, the character c_4 matches three words $w_{3,4}$ ("青海"), $w_{4,5}$("海南"), and $w_{3,4,5,6}$("青海南路"). Therefore, the word sets can be constructed by {B, M, E, S} and hence the final fusion embedding of corresponding character can be computed by the following formula::

$$v_f^{c4} \leftarrow \frac{1}{4}[v_s \oplus v_b \oplus v_m \oplus v_e]. \quad (6)$$

Here, v_b, v_m, and v_e are vectors obtained by adding all word vectors in the corresponding group of the BMES set, and v_s is the character vector from embedding lookup table. v_f^{c4} is the final fusion embedding vector of character c_4.

3.1.3 Glyph Embedding

The previous work (Meng et al., 2019) had shown that directly using deep CNNs in Chinese NER results in inferior performance. Since the Chinese character images are significantly smaller than ImageNet images (800×600) and there are only about 10,000 distinct Chinese characters, they lack training examples.

We design a specific CNN structure to extract glyph features from character images. Moreover, we select four different types of fonts for each character – *Song, LiShu, CaoShu* and *ZhuanShu*, whose each image size is 24×24 and pixel value ranges from 0 to 255. Our motivation for choosing fonts is to use as many different writing styles as possible to help to capture more features. As we can see in Figure 3, the input feature map first goes through a convolution layer with kernel size 5×5, and its output has 1024 channels. Next, a max-pooling with kernel size 4×4 is used to reduce the feature map size. Then the feature map goes through another pair convolution and max-pooling layer both with kernel sizes 2×2. After that, we convert the 2×2×512 output to a 2048 vector. Finally, the flattened vector is fed to an FC layer in order to build the glyph embedding.

3.1.4 Pinyin Embedding

The same Chinese character may have different semantic meanings. Therefore, pinyin embedding can avoid this situation. We use open-sourced

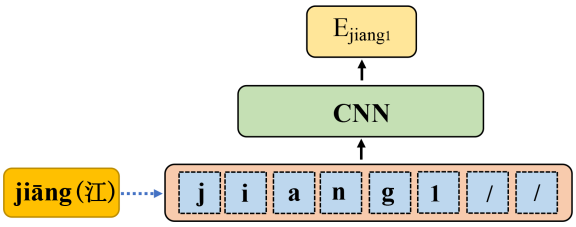

Figure 4: The process of pinyin embedding generation.

pypinyin package[4] to generate the corresponding character's pinyin sequence. As shown in Figure 4, we employ a CNN model with width 2 for each Chinese character's pinyin sequence, which is followed by average-pooling to obtain the pinyin embedding. Especially, we append the tones at the end of the character's pinyin sequence as a unique token. The length of the input pinyin sequence is fixed at 8, When the actual length of a pinyin sequence does not reach 8, the remaining slots are filled with a special letter "/".

3.2 Cross-attention Fusion Layer

After generating the character, glyph, pinyin and lexicon embeddings from the above step, we use four cross-attention encoders to fuse them. We first get the input Q_i, K_i, V_i by linear transformation of character, glyph, pinyin and lexicon embeddings:

$$\begin{bmatrix} Q_i \\ K_i \\ V_i \end{bmatrix}^\top = E_i \begin{bmatrix} W_{i,Q} \\ I_i \\ W_{i,V} \end{bmatrix}^\top, \quad (7)$$

where E_i is the embedding from character, glyph, pinyin and lexicon. W_i is the learnable parameter, and I_i denotes the identity matrix. Then we calculate the attention scores as follows:

$$A_{x,i} = (Q_{x,i} + \theta_i)^\top E_y, \quad (8)$$

$$Att_x(A_y, V_x) = Softmax(A_y)V_x, \quad (9)$$

$$Att_c(A_g, V_c) = Softmax(A_g)V_c, \quad (10)$$

$$Att_g(A_c, V_g) = Softmax(A_c)V_g, \quad (11)$$

$$Att_l(A_p, V_l) = Softmax(A_p)V_l, \quad (12)$$

$$Att_p(A_l, V_p) = Softmax(A_l)V_p, \quad (13)$$

where θ_i is the learnable parameter for attention bias, $x, y \in \{C, G, L, P\}$ represent different embeddings, A, Q, E are attention score, query vector and embedding in Eq.(8), and Att_x is cross-attention under y's attention score in Eq.(9). Eqs.(9)-(13) are the cross-attentions for the implementation of Figure 1. We can implement other cross-attentions in a similar way from Eq.(9).

Finally, we directly concatenate all features and input them into a fully connected layer for information fusion:

$$F_e = (Att_c \oplus Att_g \oplus Att_l \oplus Att_p)W_o + b, \quad (14)$$

where $\oplus$ represents the concatenation operation, W_o and b are the learnable parameters.

[4] https://pypi.org/project/pypinyin/.

Models	P	R	F1
Yang et al.(2016)$^{\xi}$	65.59	71.84	68.57
Yang et al.(2016)$^{\xi*\dagger}$	72.98	**80.15**	**76.40**
Che et al.(2013)$^{\xi*}$	77.71	72.51	72.51
Wang et al.(2013) $^{\xi*}$	76.43	72.32	72.32
Zhang et al.(2018)$^{\xi\spadesuit}$	**78.62**	78.62	75.77
Zhang et al.(2018)$^{\alpha\spadesuit}$	73.36	70.12	71.70
Lattice-LSTM	76.35	71.56	73.88
PLTE	76.78	72.54	76.40
LR-CNN	76.40	72.60	74.45
FLAT	-	-	76.45
LGN	76.13	73.68	74.89
SoftLexicon	77.28	74.07	75.64
+ bichar	77.13	75.22	76.16
MECT	77.57	76.27	76.92
CLGP+LSTM	**78.39**	**77.63**	**77.49**
BERT-Tagger	76.11	79.96	77.85
Glyce	80.87	80.40	80.62
ChineseBERT	80.77	**83.65**	82.18
MECT+BERT	-	-	82.57
CLGP(LSTM)+BERT	**82.19**	82.35	**82.83**

Table 3: Results on OntoNotes 4.0 (%), where 'ξ' denotes gold segmentation and 'α' represents the auto segmentation.

Finally, we directly concatenate all features and input them into a fully connected layer for information fusion:

$$F_e = (Att_c \oplus Att_g \oplus Att_l \oplus Att_p)W_o + b, \quad (14)$$

where $\oplus$ represents the concatenation operation, W_o and b are the learnable parameters.

3.3 Sequence Modeling and Decoding Layer

For *CLGP*, the different application scenarios can be modeled with different models of sequence. In this paper, we mainly utilize a Bi-LSTM to implement sequence modeling. And a standard CRF (John et al., 2001) layer is utilized to capture the dependencies between successive labels.

4 Experiments

We carry out extensive experiments to assess the effectiveness of our *CLGP*. Standard precision (*P*), recall (*R*), and *F1* scores are used as evaluation metrics. We evaluate our model on four

Models	P	R	F1
Chen et al. (2006)	91.22	81.71	86.20
Zhang et al. (2006)*	**92.20**	90.18	**91.18**
Zhou et al. (2013)	91.86	88.75	90.28
Lu et al. (2016)	-	-	87.94
Dong et al. (2016)	91.28	**90.62**	90.95
Lattice-LSTM	93.57	92.79	93.18
PLTE	94.25	92.30	93.26
LR-CNN	94.50	92.93	93.71
FLAT	-	-	94.12
LGN	94.19	92.73	93.46
SoftLexicon	94.63	92.70	93.66
+ bichar	94.73	93.40	94.06
MECT	94.55	94.09	94.32
CLGP+LSTM	**95.03**	**94.65**	**95.17**
BERT-Tagger	93.41	94.15	93.62
Glyce	95.57	95.51	95.07
GLYPH	-	-	95.07
MECT+BERT	-	-	96.24
CLGP(LSTM)+BERT	**96.29**	**95.84**	**96.37**

Table 4: Results on MSRA (%).

Models	NE	NM	Overall
Peng et al.(2015)	51.96	61.05	56.05
Peng et al.(2016)*	**55.28**	**62.97**	**58.99**
He et al.(2016)	50.60	59.32	54.82
He et al.(2017)*	54.50	62.17	58.23
Cao et al.(2018)	54.34	57.35	58.70
Lattice-LSTM	53.04	62.25	58.79
PLTE	53.55	64.90	59.76
LR-CNN	57.14	66.67	59.92
FLAT	-	-	60.32
LGN	55.34	64.98	60.21
SoftLexicon	59.08	62.22	61.42
+bichar	58.12	64.20	59.81
MECT	**61.91**	62.51	63.30
CLGP+LSTM	61.57	**63.37**	**64.29**
BERT-Tagger	65.69	62.21	63.74
Glyce	67.68	67.71	67.70
GLYPH	-	-	69.20
ChineseBERT	68.75	**72.97**	70.80
MECT+BERT	-	-	70.43
CLGP(LSTM)+BERT	**71.33**	70.59	**71.63**

Table 5: Results on Weibo (%). NE, NM and Overall represent F1 scores of named entities, nominal entities, and both, respectively.

Chinese NER datasets, including Ontonotes 4.0 (Weischedel et al., 2011) , MSRA (Levow, 2006), Weibo NER (Peng and Dredze, 2015; He and Sun, 2017a), and Resume (Zhang and Yang, 2018).

4.1 Implementation Details

In this paper, the most implementation details of lexicon embedding follow SoftLexicon (Ma et al., 2020), including embedding initialization, character, word embedding sizes, dropout, and the number of sequence modeling layers. And we set AdamW (Loshchilov and Hutter, 2017) optimizer to 0.05 for the Weibo dataset and 0.015 for other datasets.

4.2 Results & Analysis

Tables 3-6[5]show the experimental results of our method. Each table is divided into three blocks. The first block reports classical Chinese NER methods. The second one shows the SOTA results obtained by lexicon enhanced approaches recently. The last one is the results combining with BERT (Devlin et al., 2018). We use BiLSTM for sequence modeling and a standard CRF layer to capture the dependencies between successive labels.

OntoNotes. Table 3 shows the results on OntoNotes 4.0 dataset, where the 'ξ' denotes gold segmentation and the symbol 'α' represents automated segmentation.The other methods have no segmentation and apply lexical matching. From the table, we can draw several conclusions. *Firstly*, when combining BERT method replaces a lexicon fusion approach, the F1 score increases from 77.49% to 82.83%, which shows that combining with BERT can effectively enhance the F1 score on this dataset. *Second*, we can observe that *CLGP* achieves a relatively high precision rate, recall rate, and F1 score. *Finally*, *CLGP* can obtain the highest F1 value, which outperforms MECT by 0.57%.

MSRA. The results obtained on MSRA are shown in Table 4. In this table, the first block results are from (Dong et al., 2016). Compared to other methods, we find that the F1 score of our *CLGP* is higher than the previous SOTA methods by 0.85% without BERT. Besides, both precision and recall rate also achieve higher performances.

Weibo. From the Table 5, our method can achieve SOTA performance in F1 scores Overall. Compared with the ChineseBERT, *CLGP+BERT* improves 0.83%. And the performance of named entities (NE) also yields the highest F1 score.

Resume. The first block results of Table 6 came

[5]Table 3-6, $*$ indicates the uses of external labeled data. † denotes the model uses discrete features.

Models	P	R	F1
Zhang et al.(2018)♣	93.72	93.44	93.58
Zhang et al.(2018)♠	94.07	94.07	94.24
Zhang et al.(2018)◇	93.66	93.31	93.48
Zhang et al.(2018)♡	**94.53**	**94.29**	**94.41**
Lattice-LSTM	94.81	94.11	94.46
PLTE	95.34	95.46	95.40
LR-CNN	95.37	94.84	95.11
FLAT	-	-	95.45
LGN	95.28	95.46	95.37
SoftLexicon	95.30	95.77	95.53
+bichar	95.71	95.77	95.74
MECT	**96.40**	95.39	95.89
CLGP+LSTM	96.34	**96.69**	**96.63**
BERT-Tagger	94.79	96.53	95.71
Glyce	**96.62**	96.48	96.54
GLYPH	-	-	95.61
MECT+BERT	-	-	95.89
CLGP(LSTM)+BERT	96.57	**96.54**	**96.68**

Table 6: Results on Resume (%). To categorize the Zhang and Yang (2018) different experimental settings, we use ♣ to represents 'word-based LSTM', ♠ to denotes 'word-based+char+bichar LSTM', ◇ to indicates 'char-based LSTM', and ♡ to stands for the 'char-based +bichar+softword LSTM' model.

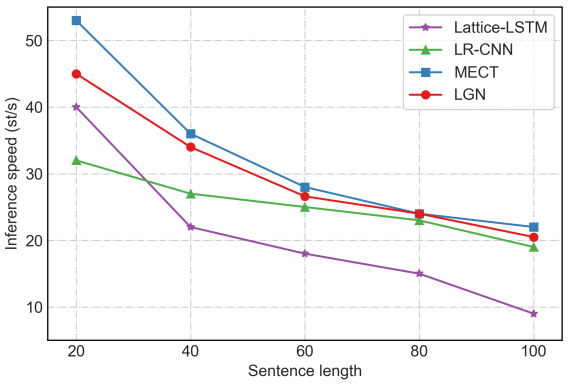

Figure 5: The inference speed of *LGN* is compared with other baselines under different sentence lengths.

Fusion	ontonote	msra	weibo	resum
Concat	76.59	94.52	63.91	95.63
+BERT	**81.70**	**95.13**	**71.54**	**96.72**
Self-Att	77.32	94.73	62.30	96.21
CG+LP	77.49	**95.17**	**64.29**	**96.63**
CL+GP	**77.89**	94.72	63.50	95.89
CP+GL	76.28	93.62	63.33	95.65
Average	76.22	94.50	63.71	96.05

Table 7: Ablation studies on different feature fusion methods, including concat, self-attention, and cross-attention between different embeddings.

from Zhang and Yang (2018) different implementations of character-level and word-level models. Different from those methods, our model integrates multiple features by cross-attention network to enhance Chinese NER. From the Table 6, the performance of our CLGP is better than other baselines.

As shown in Figure 5, we also conducted an experiment to evaluate the inference speed of sentences with different lengths. In this experiment, we set the batch size of all models to 1. The results show that CLGP has a great speedup compared with Lattice-LSTM and LR-CNN when the sentences are short. Compared with MECT model, our model needs more embeddings and more complicated fusion networks. Therefore, MECT has advantages in processing short sentences, but there is no obvious difference between CLGP with the increase of sentence length. In conclusion, CLGP has relatively better computational efficiency.

4.3 Ablation Study

We conducted two sets of ablation studies to demonstrate the effectiveness of our model. One is that different information fusion methods are introduced. The other is that the specific embedding is removed to verify its effectiveness.

4.3.1 Effectiveness of Cross-Attention

As shown in Figure 6, we validate the effects of cross-attention for *CLGP*. First, we preform concat directly for all embeddings. Then, we execute the self-attention for each embedding. Finally, We use Eq.(8) and Eq.(9) to calculate the cross attention of all other combinations to verify the effectiveness of its mechanism.

(1) **Concat Fusion**. We first directly concatenate all the features as a baseline. From Table3-7 we observe that compared with *SoftLexicon*, the F1 scores on the four datasets are higher than *SoftLexicon*, our method has obvious performance advantages. This reveals that glyph features and pinyin information can significantly enhance Chinese NER. In addition, we add BERT embedding on the basis of concat, which also has obvious advantages compared with *GLYPH* based on *glyph + Bert* structure. This further illustrates that incorporating lexicon information is critical for Chinese NER.

(2) **Self-Attention Fusion**. We adapt the fusion network into four sets of self-attention without

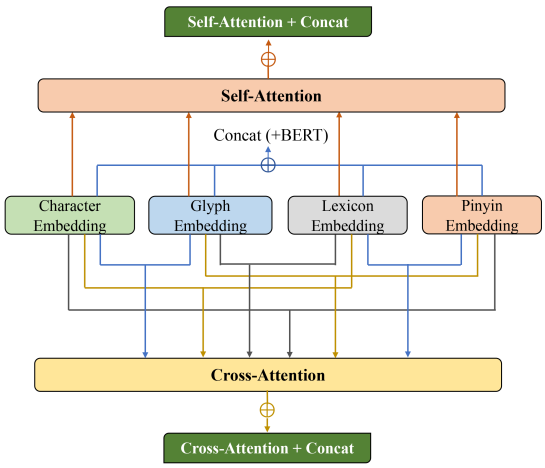

Figure 6: The ablation study to assess the effectiveness of cross-attention in our model.

Reduce	otnt4.0	msra	weibo	resum
-Char	**77.13**	**94.85**	**63.59**	95.48
-Lexicon	75.97	93.20	61.80	94.39
-Glyph	76.83	94.83	61.59	95.14
-Pinyin	76.45	94.61	62.47	**95.67**
-Chars,Lex	73.64	93.91	61.06	94.36
-Chars,Gly	74.56	93.52	61.73	94.53
-Chars,Piy	**75.35**	**94.39**	62.46	**95.74**
-Lex,Piy	74.75	93.73	61.31	94.25
-Gly,Piy	75.29	94.36	**62.49**	94.39
-Gly,Lex	71.83	93.38	61.04	94.17

Table 8: Ablation studies that remove different embeddings to verify their effectiveness.

modifying the query vector. As shown in Table 7, the F1 scores are higher than the direct concat, except for the Weibo dataset. However, the performance is worse than the best combination in cross-attention but better than the average.

(3) **Cross-Attention Fusion**. We studied all possible embedding combinations in cross-attention. From the second block of Table 7, we find that the F1 difference between each combination and the average is small. But there is still an obvious gap between the best combination and the worst combination. However, the best-combined performance of cross-attention is obviously better than the performances of concat and self-attention.

4.3.2 Effectiveness of Multi-Feature

We design a specific architecture to fuse the features, it is necessary to demonstrate the impact of different embeddings on our model.

(1) **Remove an Embedding**. In the first block of Table 8, we remove one embedding. Two embeddings perform cross-attention with the best combination while the last one directly uses self-attention. As can be seen in Table 8, removing the character embedding has the least impact, while removing the lexicon has a great impact on our model.

(2) **Remove two Embeddings**. In the second block of Table 8, two embeddings are removed, and the rest employs cross-attention. The results show that embedding without Pinyin and Characters has the least impact on performance.

models	otnt	msra	weibo	resm
LSTM	**77.49**	**95.47**	**64.29**	**96.63**
CNN	73.72	95.42	62.68	95.34
Transformer	71.81	93.40	60.93	95.13

Table 9: The F1 scores of different sequence models. Otnt and resm represent ontonotes 4.0 and resume datasets, respectively.

4.4 Scalability Study

Table 9 shows the results obtained with different sequence models in *CLGP*. As we can see, the model based on LSTM yields the best performance.

4.5 Combining BERT

The last blocks of Tables 3-6 are the results obtained by combining BERT [6]. We follow Ma et al. (2020) by using a BERT encoder to obtain the contextual representations of each sequence and concatenate them into final embedding. The results show that combining BERT outperform the BERT tagger on all four datasets. Therefore, this result shows that *CLGP* can be effectively integrated with the pre-trained models, such as BERT.

5 Conclusion

This paper proposes *CLGP*, which uses the embedding scheme to preserve the lexicon matching, and two specific CNN architectures to extract glyph and pinyin embeddings, then combine the four embeddings by cross-attention based network to enhance the expressiveness of the Chinese NER task. The experimental results on four Chinese public NER datasets show that *CLGP* sets state-of-the-art performance. We also conducted a series of ablation experiments to demonstrate the effectiveness of our method.

[6] https://github.com/google-research/bert.

References

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.

F. Dai and Z. Cai. 2017. Glyph-aware embedding of chinese characters. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 64–69.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ruixue Ding, Pengjun Xie, Xiaoyan Zhang, Wei Lu, Linlin Li, and Luo Si. 2019. A neural multi-digraph model for chinese ner with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1462–1467.

C. Dong, J. Zhang, C. Zong, M. Hattori, and H. Di. 2016. Character-based lstm-crf with radical-level features for chinese named entity recognition. In *Natural Language Understanding and Intelligent Applications*, pages 239–250. Springer.

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. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1040–1050.

H. He and X. Sun. 2017a. F-score driven max margin neural network for named entity recognition in chinese social media. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 713–718.

H.; He and X. Sun. 2017b. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

J.; He and H. Wang. 2008. Chinese named entity recognition and word segmentation based on character. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

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

Y. Jie, Z. Teng, M. Zhang, and Z. Yue. 2016. Combining discrete and neural features for sequence labeling. In *International Conference on Intelligent Text Processing Computational Linguistics*.

D. John, M. Andrew, and C. Fernando. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. 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*.

G. Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117.

H. Li, M. Hagiwara, Q. Li, and H. Ji. 2014. Comparison of the impact of word segmentation on name tagging for chinese and japanese. In *LREC*, pages 2532–2536.

X. Li, H. Yan, X. Qiu, and X. Huang. 2020. Flat: Chinese ner using flat-lattice transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6836–6842.

Y. Li, W. Li, F. Sun, and S. Li. 2015. Component-enhanced chinese character embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 829–834.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning character-level compositionality with visual features. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2059–2068, Vancouver, Canada. Association for Computational Linguistics.

L. Liu, J. Shang, F. Xu, R. Xiang, and J. Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Wei Liu, Tongge Xu, Qinghua Xu, Jiayu Song, and Yueran Zu. 2019. An encoding strategy based word-character lstm for chinese ner. 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 2379–2389.

Z. Liu, C. Zhu, and T. Zhao. 2010. Chinese named entity recognition with a sequence labeling approach: based on characters, or based on words? In *International Conference on Intelligent Computing*, pages 634–640. Springer.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ruotian Ma, Minlong Peng, Qi Zhang, Zhongyu Wei, and Xuan-Jing 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.

X. Ma and E. 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)*.

Y. Meng, W. Wu, F. Wang, X. Li, P. Nie, F. Yin, M. Li, Q. Han, X. Sun, and J. Li. 2019. Glyce: glyph-vectors for chinese character representations. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 2746–2757.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, volume 26, pages 3111–3119.

N. Peng and M. 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.

Arijit Sehanobish and Chan Hee Song. 2020. Using chinese glyphs for named entity recognition. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 13921–13922.

Y. Shao, C. Hardmeier, Jörg T., and Joakim N. 2017. Character-based joint segmentation and pos tagging for chinese using bidirectional rnn-crf. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 173–183.

X. Shi, J. Zhai, X. Yang, Z. Xie, , and C. Liu. 2015. Radical embedding: Delving deeper to chinese radicals. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 594–598.

Tzu-Ray Su and Hung-Yi Lee. 2017. Learning chinese word representations from glyphs of characters. In *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 264–273.

D. Sui, Y. Chen, K. Liu, J. Zhao, and S. Liu. 2019. Leverage lexical knowledge for chinese named entity recognition via collaborative graph network. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3830–3840.

Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2014. Radical-enhanced chinese character embedding. In *In International Conference on Neural Information Processing*, pages 279–286. Springer.

Z. Sun, X. Li, X. Sun, Y. Meng, X. Ao, F. He, Q.and Wu, and J. Li. 2021. Chinesebert: Chinese pretraining enhanced by glyph and pinyin information. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

R. Weischedel, S. Pradhan, L. Ramshaw, M. Palmer, N. Xue, M. Marcus, A. Taylor, C. Greenberg, E. Hovy, and R. Belvin. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.

Shuang Wu, Xiaoning Song, and Zhenhua Feng. 2021a. 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.

Shuang Wu, Xiaoning Song, Qi Zhang, Minlong Peng, and Zhenhua Feng. 2021b. Mect: Multi-metadata embedding based cross-transformer for chinese named entity recognition. In *The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*.

Z. Xuan, R. Bao, and S. Jiang. 2020. Fgn: Fusion glyph network for chinese named entity recognition. *arXiv preprint arXiv:2001.05272*.

R. Yin, Q. Wang, P. Li, R. Li, and B. Wang. 2016. Multi-granularity chinese word embedding. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 981–986.

Xiang Zhang and Yann LeCun. 2017. Which encoding is the best for text classification in chinese, english, japanese and korean?

Y. Zhang and J. 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.