華東師範大學
EAST CHINA NORMAL UNIVERSITY

Works motivated by Lattice-LSTM

浦天岭

Lattice-LSTM

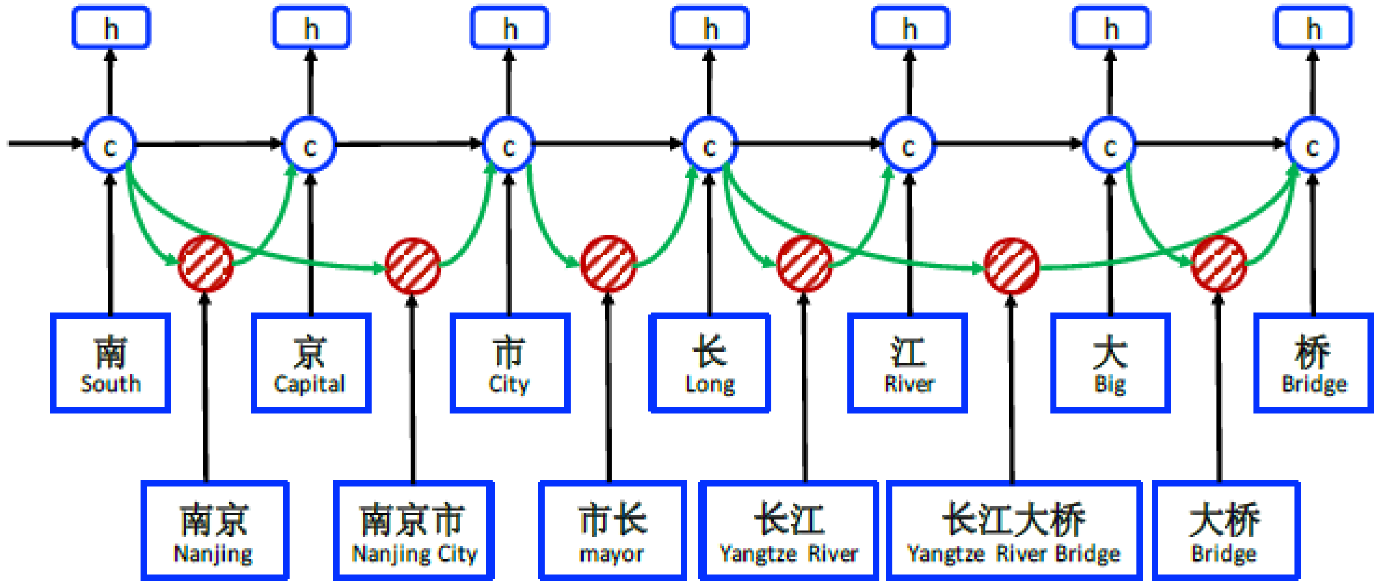
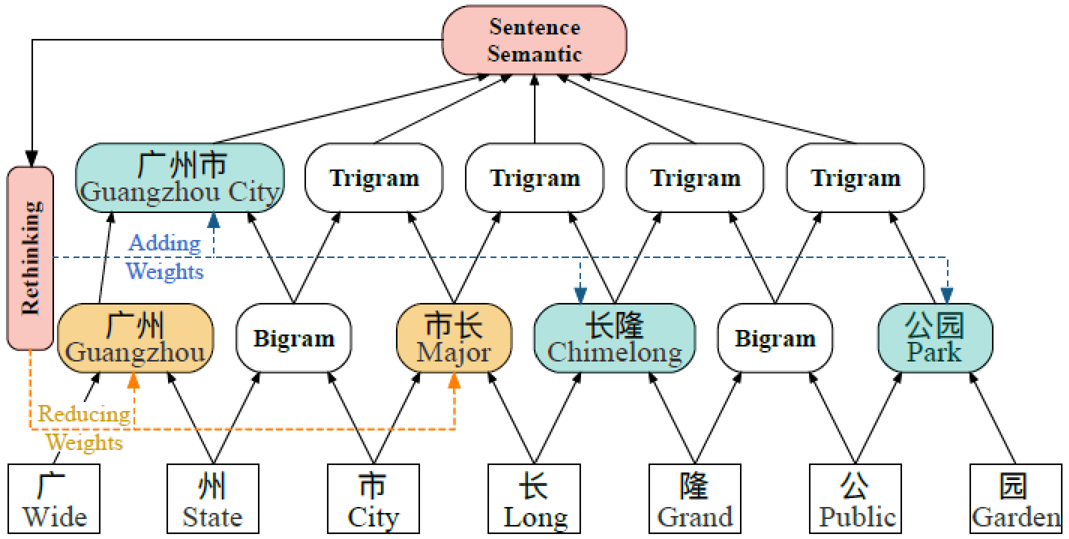


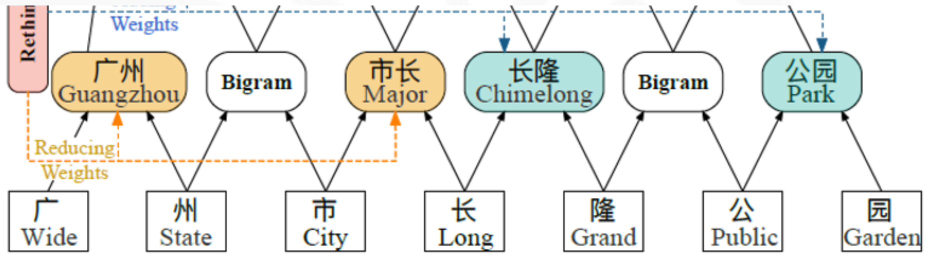
Figure 2: Lattice LSTM structure.

CNN-Based Chinese NER with Lexicon Rethinking (IJCAI2019)

Tao Gui[1*], **Ruotian Ma**[1*], **Qi Zhang**[1], **Lujun Zhao**[1], **Yu-Gang Jiang**[1,2] and **Xuanjing Huang**[1]

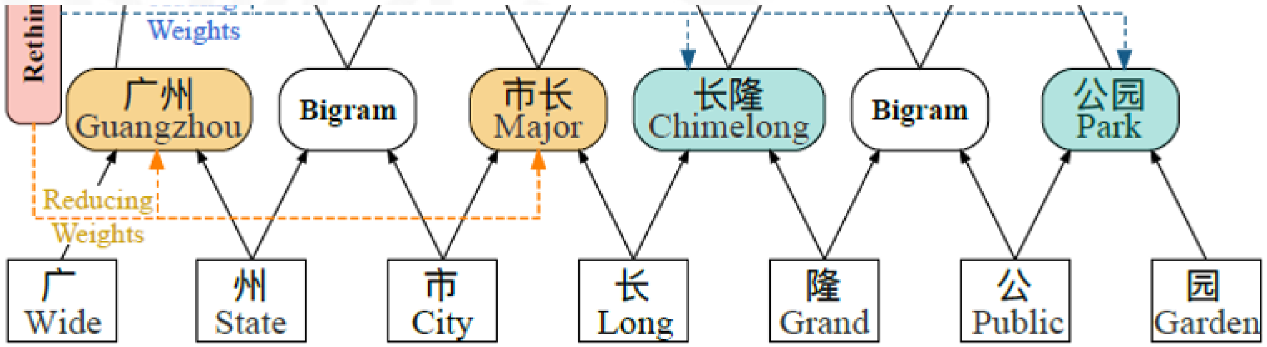
[1]School of Computer Science, Fudan University, Shanghai, China
[2]Jilian Technology Group (Video++), Shanghai, China
{tgui16, rtma15, qz, ljzhao16, ygj, xjhuang}@fudan.edu.cn





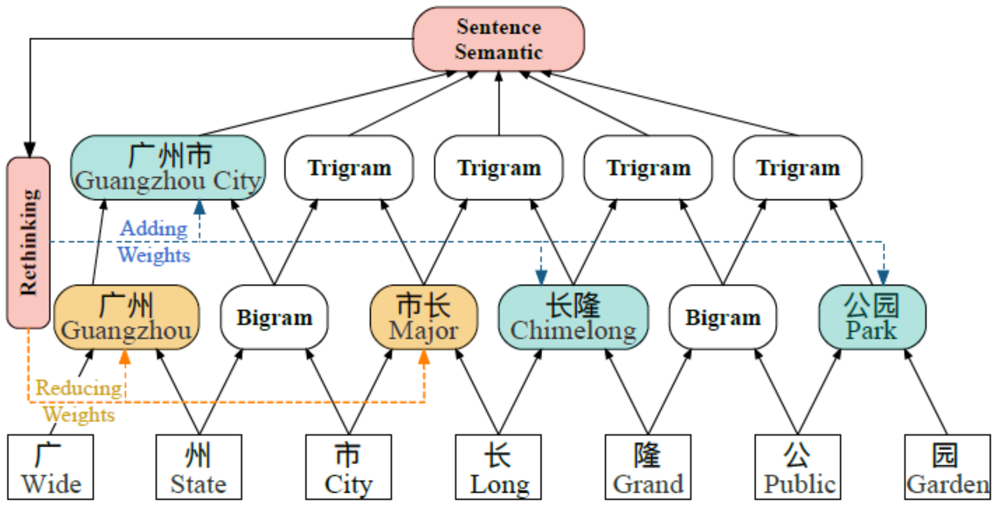
lexicon features. The CNNs apply multiple filters $\mathbf{H} \in \mathbb{R}^{d \times 2}$ with a window size of 2 to obtain the bigram information and stack the multiple layers to obtain the multigram information, as follows:

$$\begin{aligned} \mathbf{C}_m^2 &= \tanh(\langle \mathbf{C}^1[*, m:m+1], \mathbf{H}_1 \rangle + b_1) \\ \mathbf{C}_m^l &= \tanh(\langle \mathbf{C}^{l-1}[*, m:m+1], \mathbf{H}_{l-1} \rangle + b_{l-1}), \end{aligned} \quad (1)$$



From the perspective of the combination of characters, the l gram feature $\mathbf{C}_m^l$ corresponds to the word $\mathbf{w}_m^l$ (Figure 2). To incorporate the lexicon feature effectively, we use the vector-based attention [Chen *et al.*, 2018] to combine the l gram feature with the word feature, as follows:

$$\begin{aligned}
\mathbf{i}_1 &= \sigma(W_i \mathbf{C}_m^l + U_i \mathbf{w}_m^l + b_i) \\
\mathbf{f}_1 &= \sigma(W_f \mathbf{C}_m^l + U_f \mathbf{w}_m^l + b_f) \\
\mathbf{u}_1 &= \tanh(W_u \mathbf{C}_m^l + U_u \mathbf{w}_m^l + b_u) \qquad (2) \\
\mathbf{i}_1', \mathbf{f}_1' &= \mathrm{softmax}(\mathbf{i}_1, \mathbf{f}_1) \\
X_m^l &= \mathbf{i}_1' \odot \mathbf{u}_1 + \mathbf{f}_1' \odot \mathbf{C}_m^l,
\end{aligned}$$



In this work, we treat the features at the top layer of CNN, X_m^L, as the high-level features. We then use these features to readjust the weights of the lexicon attention module by adding a feedback layer to each CNN layer, as follows:

$$\begin{aligned}
\mathbf{i}_2 &= \sigma(W_i^* \mathbf{C}_m^l + U_i^* \mathbf{w}_m^l + V_i X_m^L + b_i^*) \\
\mathbf{f}_2 &= \sigma(W_f^* \mathbf{C}_m^l + U_f^* \mathbf{w}_m^l + V_f X_m^L + b_f^*) \\
\mathbf{r}_2 &= \sigma(W_r \mathbf{C}_m^l + U_r \mathbf{w}_m^l + V_r X_m^L + b_r) \\
\mathbf{u}_2 &= \tanh(W_u^* \mathbf{C}_m^l + U_u^* \mathbf{w}_m^l + b_u^*) \\
\mathbf{i}_2', \mathbf{f}_2', \mathbf{r}_2' &= \mathrm{softmax}(\mathbf{i}_2, \mathbf{f}_2, \mathbf{r}_2) \\
X_m^{l'} &= \mathbf{i}_2' \odot \mathbf{u}_2 + \mathbf{f}_2' \odot \mathbf{C}_m^l + \mathbf{r}_2' \odot X_m^L,
\end{aligned} \tag{3}$$

Through the CNN encoder and lexicon attention module, the model obtains the respective l gram features X_m^l within each layer of the model. These feature values correspond to the multi-scale l-grams, of which some are redundant (i.e., unigram X_m^1, bigram X_m^2, $\ldots$, and L-gram X_m^L). In this work, we propose the use of the multi-scale feature attention [Wang *et al.*, 2018] to adaptively selects the features of different scales in each position of a sentence, as follows:

$$s_m^l = \sum_{d=1}^{D} X_m^{l'}[d], \quad \alpha_m^l = \frac{\exp(s_m^l)}{\sum_{l=1}^{L} \exp(s_m^l)} \qquad (4)$$
$$X_m^{att} = \sum_{l=1}^{L} \alpha_m^l X_m^{l'}.$$

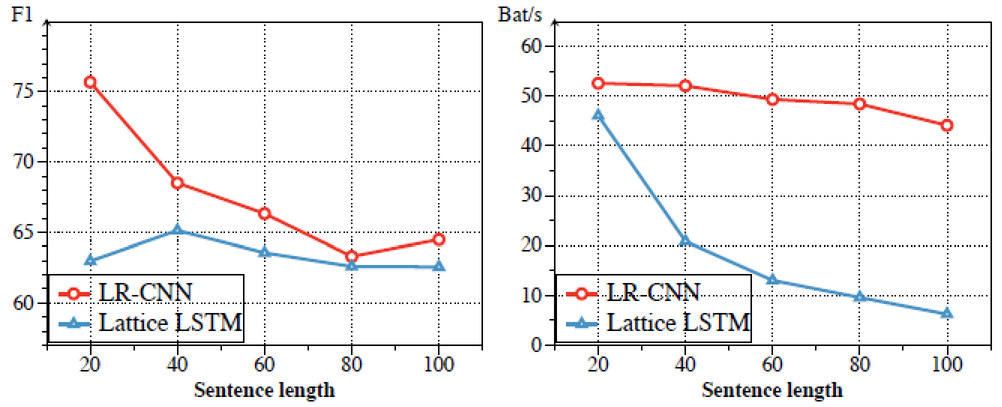


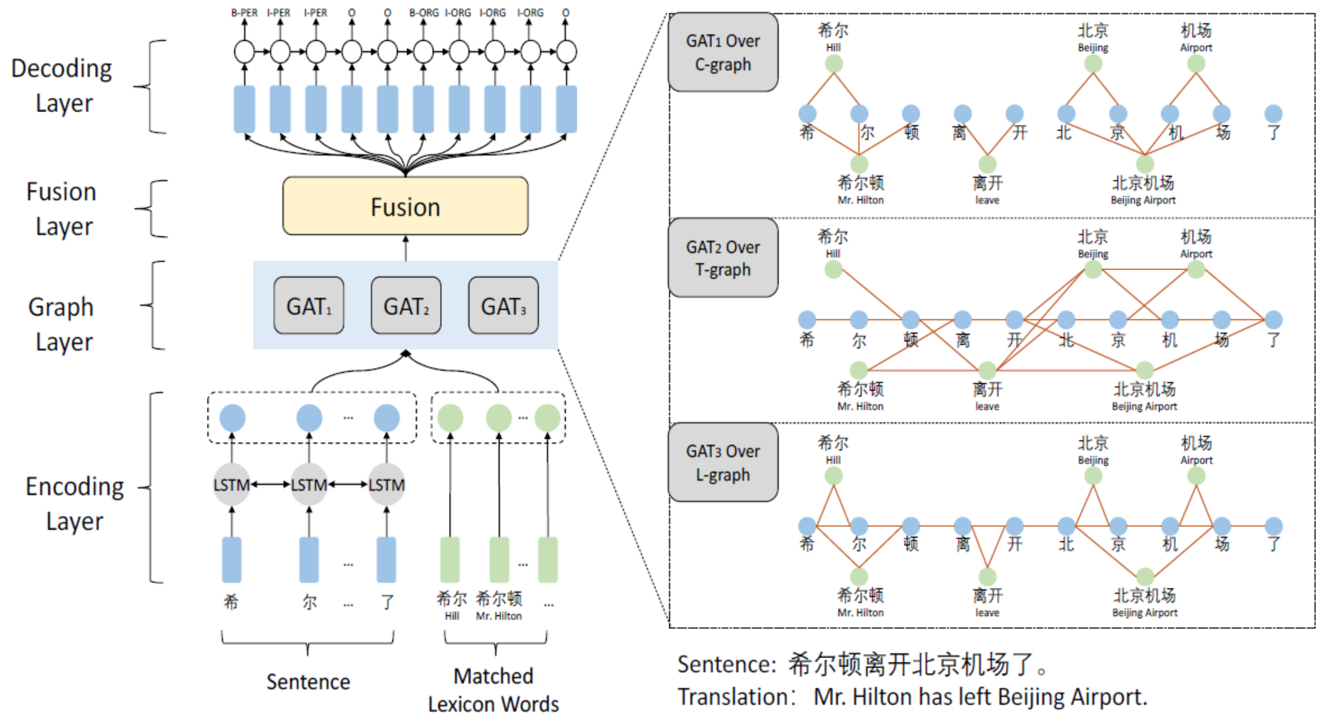
Figure 3: F1 and speed against sentence length. Bat/s refers to the number of batches can be processed per second.

Models	OntoNotes	MSRA	Weibo	Resume
LR-CNN	74.45	93.71	59.92	95.11
- Rethink	73.09	93.58	57.86	95.04
- Lexicon	65.31	86.81	49.58	94.27
- Lexicon - Rethink	59.86	87.81	50.75	94.47

Leverage Lexical Knowledge for Chinese Named Entity Recognition via Collaborative Graph Network

(EMNLP2019)

Dianbo Sui[1,2], Yubo Chen[1], Kang Liu[1,2], Jun Zhao[1,2], Shengping Liu[3]
[1] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Beijing Unisound Information Technology Co., Ltd, Beijing, 100028, China
{dianbo.sui, yubo.chen, kliu, jzhao}@nlpr.ia.ac.cn
liushengping@unisound.com



Sentence: 希尔顿离开北京机场了。
Translation: Mr. Hilton has left Beijing Airport.

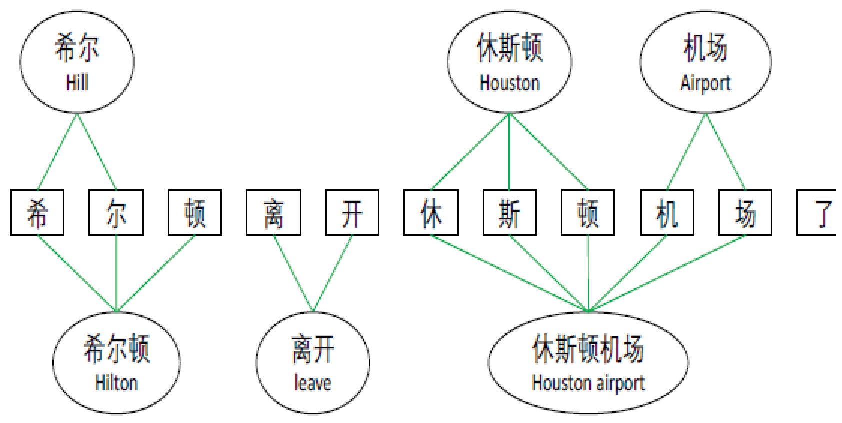


Figure 2: Word-Character Containing graph

Word-Character Containing graph

With the C-graph, the characters in the sentence can capture the boundaries and semantic information of self-matched lexical words. As shown in Figure 2, if a lexical word i contains a character j, the (i, j)-entry of the C-graph corresponding adjacency matrix $\mathbf{A}^C$ will be assigned a value of 1.

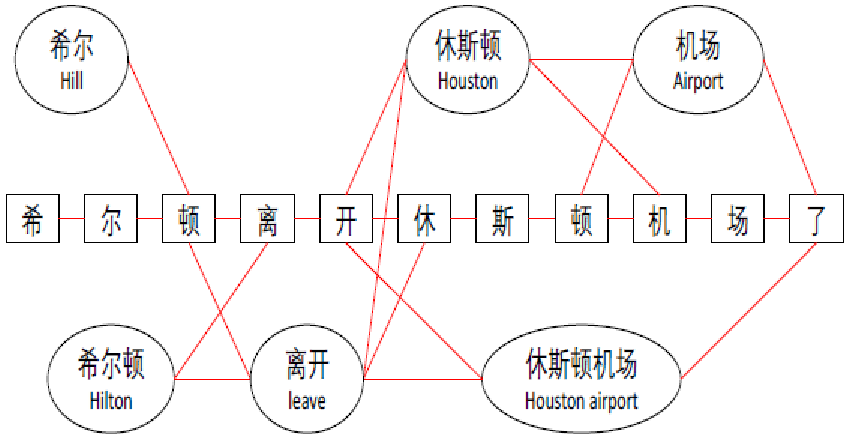


Figure 3: Word-Character Transition graph

The T-graph is to assist the character to capture the semantic information of the nearest contextual lexical words. As shown in Figure 3, if a lexical word i or a character m matches the nearest preceding or following subsequence of a character j, the (i,j) or (m,j)-entry of the T-graph corresponding adjacency matrix $\mathbf{A}^T$ will be assigned a value of 1. Moreover, for capturing the context relation between lexical words, if a lexical word i is the preceding or following context of another lexical word k, we will assign "$\mathbf{A}^T_{ik} = 1$". Note that the T-graph is the same with the word cutting graph which is used in Chinese Word Segmentation.

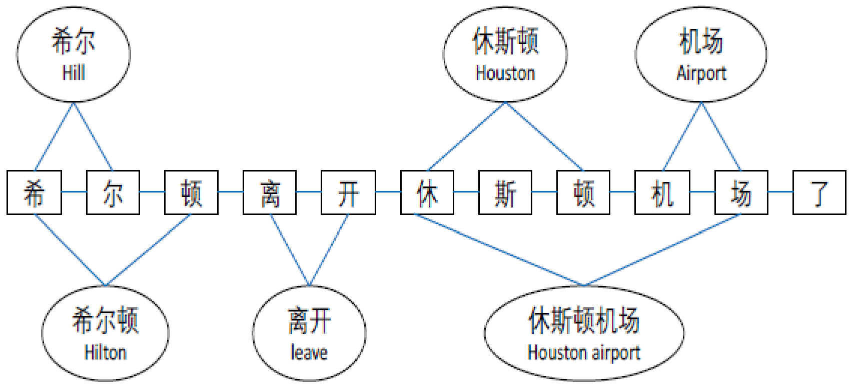


Figure 4: Word-Character Lattice graph

Zhang and Yang (2018) propose a lattice structure LSTM to exploit lexical knowledge for Chinese NER. A lattice structure can capture the information of the nearest contextual lexical words implicitly and capture some information of self-matched lexical words. As shown in Figure 4, if a character m is the nearest preceding or following character of a character j, the (m, j)-entry of the L-graph corresponding adjacency matrix $\mathbf{A}^L$ will be assigned a value of 1. Moreover, if a character j matches the lexical word i first character or end character, we will assign "$\mathbf{A}^L_{ij} = 1$".

$$\mathbf{Node}_f = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n, \mathbf{wv}_1, \mathbf{wv}_2, ..., \mathbf{wv}_m]$$

We use Graph Attention Networks (GAT) to model over three interactive graphs. In an M-layer GAT, the input of j-th layer is a set of node features, $\mathbf{NF}^j = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N\}$, together with an adjacency matrix $\mathbf{A}$, $\mathbf{f}_i \in \mathbb{R}^F$, $\mathbf{A} \in \mathbb{R}^{N\times N}$, where N denotes the number of the nodes and F is the the dimension of features at j-th layer. The output of j-th layer is a new set of node features, $\mathbf{NF}^{(j+1)} = \{\mathbf{f}'_1, \mathbf{f}'_2, ..., \mathbf{f}'_N\}$. A GAT operation with K independent attention head can be written as :

$$\mathbf{f}'_i = \overset{K}{\underset{k=1}{\parallel}} \sigma(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{f}_j) \tag{7}$$

$$\alpha_{ij}^k = \frac{\exp(\mathrm{LeakyReLU}(\mathbf{a}^\mathrm{T}[\mathbf{W}^\mathrm{k}\mathbf{f}_\mathrm{i} \| \mathbf{W}^\mathrm{K}\mathbf{f}_\mathrm{j}]))}{\Sigma_{k\in\mathcal{N}_i} \exp(\mathrm{LeakyReLU}(\mathbf{a}^\mathrm{T}[\mathbf{W}^\mathrm{k}\mathbf{f}_\mathrm{i} \| \mathbf{W}^\mathrm{K}\mathbf{f}_\mathrm{k}]))} \tag{8}$$

where $\|$ denotes concatenation operation, σ is a nonlinear activation function, $\mathcal{N}_i$ is the neighborhood of node i in the graph, α_{ij}^k are the attention coefficients, $\mathbf{W}^k \in \mathbb{R}^{F'\times F}$, and $\mathbf{a} \in \mathbb{R}^{2F'}$ is a single-layer feed-forward neural network. Note that, the dimension of the output $\mathbf{f}'_i$ is KF'. At the last layer, averaging will be adopted, and the dimension of final output features is F'.

$$\mathbf{f}_i^{final} = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{f}_j) \tag{9}$$

Fusion Layer

A fusion layer is used to fuse different lexical knowledge captured by word-character interactive graphs. The input of the fusion layer is the contextual representation $\mathbf{H}$ and the output of the graph layer $\mathbf{Q}_i$, $i \in \{1, 2, 3\}$. The equation of the fusion layer is introduced below:

$$\mathbf{R} = \mathbf{W}_1\mathbf{H} + \mathbf{W}_2\mathbf{Q}_1 + \mathbf{W}_3\mathbf{Q}_2 + \mathbf{W}_4\mathbf{Q}_3 \quad (14)$$

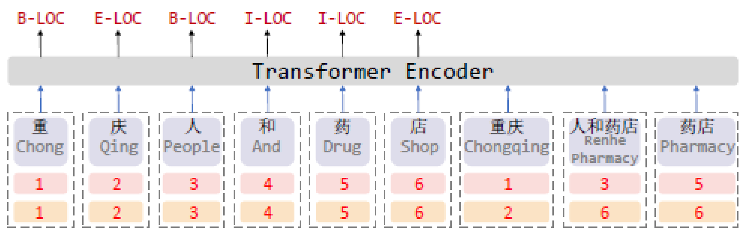
where $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$ and $\mathbf{W}_4$ are trainable matrices. Via a fusion layer, we obtain a matrix $\mathbf{R}$, $\mathbf{R} \in \mathbb{R}^{F' \times n}$, which is a new sentence representation integrating the contextual information as well as the lexical knowledge of self-matched lexical words and the nearest contextual lexical words.

華東師範大學
EAST CHINA NORMAL UNIVERSITY

Models	Dataset		
	OntoNotes	Weibo NER	MSRA
Complete model	**74.79**	**63.09**	**93.47**
w/o C	72.24	60.75	93.35
w/o T	71.57	60.94	93.02
w/o L	72.87	60.69	93.21
w/o C & T	70.53	58.51	92.72
w/o C & L	65.81	58.65	91.98
w/o T & L	71.41	58.72	92.80
BiLSTM+CRF	61.84	52.77	88.05

FLAT: Chinese NER Using Flat-Lattice Transformer (ACL20)

Xiaonan Li, Hang Yan, Xipeng Qiu*, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
lixiaonan_xdu@outlook.com, {hyan19, xpqiu, xjhuang}@fudan.edu.cn



(c) Flat-Lattice Transformer.

$$d_{ij}^{(hh)} = head[i] - head[j],$$
$$d_{ij}^{(ht)} = head[i] - tail[j],$$
$$d_{ij}^{(th)} = tail[i] - head[j],$$
$$d_{ij}^{(tt)} = tail[i] - tail[j],$$

To encode the interactions among spans, we propose the relative position encoding of spans. For two spans x_i and x_j in the lattice, there are three kinds of relations between them: intersection, inclusion and separation, determined by their heads and tails. Instead of directly encoding these three kinds of relations, we use a dense vector to model their relations. It is calculated by continuous transformation of the head and tail information. Thus, we think it can not only represent the relation between two tokens, but also indicate more detailed information, such as the distance between a character and a word. Let head[i] and tail[i] denote the head and tail position of span x_i. Four kinds of relative distances can be used to indicate the relation between x_i and x_j .

$$R_{ij} = \mathrm{ReLU}(W_r(\mathbf{p}_{d_{ij}^{(hh)}} \oplus \mathbf{p}_{d_{ij}^{(th)}} \oplus \mathbf{p}_{d_{ij}^{(ht)}} \oplus \mathbf{p}_{d_{ij}^{(tt)}})), \quad (8)$$

where W_r is a learnable parameter, $\oplus$ denotes the concatenation operator, and $\mathbf{p}_d$ is calculated as in Vaswani et al. (2017),

$$\mathbf{p}_d^{(2k)} = \sin\left(d/10000^{2k/d_{model}}\right), \quad (9)$$

$$\mathbf{p}_d^{(2k+1)} = \cos\left(d/10000^{2k/d_{model}}\right), \quad (10)$$

where d is $d_{ij}^{(hh)}$, $d_{ij}^{(ht)}$, $d_{ij}^{(th)}$ or $d_{ij}^{(tt)}$ and k denotes the index of dimension of position encoding. Then we use a variant of self-attention (Dai et al., 2019) to leverage the relative span position encoding as follows:

对于一个向量序列 $\boldsymbol{H} = [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_T] \in \mathbb{R}^{D_h \times T}$，首先用自注意力模型来对其进行编码，即

$$\text{self-att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{V}\,\text{softmax}\Big(\frac{\boldsymbol{K}^\top \boldsymbol{Q}}{\sqrt{D_k}}\Big), \tag{15.107}$$

$$\boldsymbol{Q} = \boldsymbol{W}_q\boldsymbol{H}, \boldsymbol{K} = \boldsymbol{W}_k\boldsymbol{H}, \boldsymbol{V} = \boldsymbol{W}_v\boldsymbol{H}, \tag{15.108}$$

其中 D_k 是输入矩阵 $\boldsymbol{Q}$ 和 $\boldsymbol{K}$ 中列向量的维度，$\boldsymbol{W}_q \in \mathbb{R}^{D_k \times D_h}$，$\boldsymbol{W}_k \in \mathbb{R}^{D_k \times D_h}$，$\boldsymbol{W}_v \in \mathbb{R}^{D_v \times D_h}$ 为三个投影矩阵.

$$\begin{aligned}
\mathbf{A}_{i,j}^{\text{abs}} &= \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} \\
&+ \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}.
\end{aligned}$$

$$\begin{aligned}
\mathbf{A}_{i,j}^{\text{rel}} &= \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} \\
&+ \underbrace{u^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{v^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}.
\end{aligned}$$

- The first change we make is to replace all appearances of the absolute positional embedding $\mathbf{U}_j$ for computing key vectors in term (b) and (d) with its relative counterpart $\mathbf{R}_{i-j}$. This essentially reflects the prior that only the relative distance matters for where to attend. Note that $\mathbf{R}$ is a sinusoid encoding matrix (Vaswani et al., 2017) without learnable parameters.
- Secondly, we introduce a trainable parameter $u \in \mathbb{R}^d$ to replace the query $\mathbf{U}_i^\top \mathbf{W}_q^\top$ in term (c). In this case, since the query vector is the same for all query positions, it suggests that the attentive bias towards different words should remain the same regardless of the query position. With a similar reasoning, a trainable parameter $v \in \mathbb{R}^d$ is added to substitute $\mathbf{U}_i^\top \mathbf{W}_q^\top$ in term (d).
- Finally, we deliberately separate the two weight matrices $\mathbf{W}_{k,E}$ and $\mathbf{W}_{k,R}$ for producing the content-based key vectors and location-based key vectors respectively.

Under the new parameterization, each term has an intuitive meaning: term (a) represents content-based addressing, term (b) captures a content-dependent positional bias, term (c) governs a global content bias, and (d) encodes a global positional bias.

Simplify the Usage of Lexicon in Chinese NER（ACL2020）

Minlong Peng, Ruotian Ma, Qi Zhang, Xuanjing Huang
School of Computer Science, Fudan University, Shanghai, China
{mlpeng16,rtma19,qz,xjhuang}@fudan.edu.cn

From our view, the advance of Lattice-LSTM comes from two points. The first point is that it preserve all possible matching words for each character. This can avoid the error propagation introduced by heuristically choosing a matching result of the character to the NER system. The second point is that it can introduce pre-trained word embeddings to the system, which bring great help to the final performance. While the disadvantage of Lattice-LSTM is that it turns the input form of a sentence from a chained sequence into a graph. This will greatly increase the computational cost for sentence modeling. Therefore, the design of our method should try to keep the chained input form of the sentence and at the same time, achieve the above two advanced points of Lattice-LSTM.

With this in mind, our method design was firstly motivated by the Softword technique, which was originally used for incorporating word segmentation information into downstream tasks (Zhao and Kit, 2008; Peng and Dredze, 2016). Precisely, the Softword technique augments the representation of a character with the embedding of its corresponding segmentation label:

$$\mathbf{x}_j^c \leftarrow [\mathbf{x}_j^c; \boldsymbol{e}^{seg}(seg(c_j))]. \tag{10}$$

Here, $seg(c_j) \in \mathcal{Y}_{seg}$ denotes the segmentation label of the character c_j predicted by the word segmentor, $\boldsymbol{e}^{seg}$ denotes the segmentation label embedding lookup table, and commonly $\mathcal{Y}_{seg} = \{\mathrm{B}, \mathrm{M}, \mathrm{E}, \mathrm{S}\}$ with B, M, E indicating that the character is the beginning, middle, and end of a word, respectively, and S indicating that the character itself forms a single-character word.

The first idea we come out based on the Softword technique is to construct a word segmenter using the lexicon and allow a character to have multiple segmentation labels. Take the sentence $s = \{c_1, c_2, c_3, c_4, c_5\}$ as an example. If both its sub-sequences $\{c_1, c_2, c_3, c_4\}$ and $\{c_3, c_4\}$ match a word of the lexicon, then the segmentation label sequence of s using the lexicon is $segs(s) = \{\{\mathrm{B}\}, \{\mathrm{M}\}, \{\mathrm{B}, \mathrm{M}\}, \{\mathrm{E}\}, \{\mathrm{O}\}\}$. Here, $segs(s)_1 = \{\mathrm{B}\}$ indicates that there is at least one sub-sequence of s matching a word of the lexicon and beginning with c_1, $segs(s)_3 = \{\mathrm{B}, \mathrm{M}\}$ means that there is at least one sub-sequence of s matching the lexicon and beginning with c_3 and there is also at least one lexicon matched sub-sequence in the middle of which c_3 occurs, and $segs(s)_5 = \{\mathrm{O}\}$ means that there is no sub-sequence of s that matches the lexicon and contains c_5. The character representation is then obtained by:

$$\mathbf{x}_j^c \leftarrow [\mathbf{x}_j^c; \boldsymbol{e}^{seg}(segs(s)_j)], \tag{11}$$

However, through the analysis of ExSoftword, we can find out that the ExSoftword method cannot fully inherit the two merits of Lattice-LSTM. Firstly, it cannot not introduce pre-trained word embeddings. Secondly, though it tries to keep all the lexicon matching results by allowing a character to have multiple segmentation labels, it still loses lots of information. In many cases, we cannot restore the matching results from the segmentation label sequence. Consider the case that in the sentence $s = \{c_1, c_2, c_3, c_4\}$, $\{c_1, c_2, c_3\}$ and $\{c_2, c_3, c_4\}$ match the lexicon. In this case, $segs(s) = \{\{\text{B}\}, \{\text{B}, \text{M}\}, \{\text{M}, \text{E}\}, \{\text{E}\}\}$. However, based on $segs(s)$ and s, we cannot say that it is $\{c_1, c_2, c_3\}$ and $\{c_2, c_3, c_4\}$ matching the lexicon since we will obtain the same segmentation label sequence when $\{c_1, c_2, c_3, c_4\}$ and $\{c_2, c_3\}$ match the lexicon.

華東師範大學
EAST CHINA NORMAL UNIVERSITY

Consider the sentence $s = \{c_1, \cdots, c_5\}$ and suppose that $\{c_1, c_2\}$, $\{c_1, c_2, c_3\}$, $\{c_2, c_3, c_4\}$, and $\{c_2, c_3, c_4, c_5\}$ match the lexicon. Then, for c_2, $\mathrm{B}(\mathrm{c}_2) = \{\{\mathrm{c}_2, \mathrm{c}_3, \mathrm{c}_4\}, \{\mathrm{c}_2, \mathrm{c}_3, \mathrm{c}_4, \mathrm{c}_5\}\}$, $\mathrm{M}(\mathrm{c}_2) = \{\{\mathrm{c}_1, \mathrm{c}_2, \mathrm{c}_3\}\}$, $\mathrm{E}(\mathrm{c}_2) = \{\{\mathrm{c}_1, \mathrm{c}_2\}\}$, and $\mathrm{S}(\mathrm{c}_2) = \{\mathrm{NONE}\}$. In this way, we can now introduce the pre-trained word embeddings and moreover, we can exactly restore the matching results from the word sets of each character.

$$\begin{gathered}\boldsymbol{e}^s(\mathrm{B}, \mathrm{M}, \mathrm{E}, \mathrm{S}) = [\boldsymbol{v}^s(\mathrm{B}) \oplus \mathrm{v}^s(\mathrm{M}) \oplus \mathrm{v}^s(\mathrm{E}) \oplus \mathrm{v}^s(\mathrm{S})], \\ \mathbf{x}^c \leftarrow [\mathbf{x}^c; \boldsymbol{e}^s(\mathrm{B}, \mathrm{M}, \mathrm{E}, \mathrm{S})].\end{gathered} \tag{12}$$

B, M, E, S are the sets of the corresponding words.

MP : $$\boldsymbol{v}^s(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{w \in \mathcal{S}} \boldsymbol{e}^w(w).$$

WP : $$v^s(S) = \frac{1}{Z}\sum_{w \in S} z(w)\mathbf{e}^w(w), \qquad (14)$$

where
$$Z = \sum_{w \in \mathrm{B} \cup \mathrm{M} \cup \mathrm{E} \cup \mathrm{S}} z(w).$$

SWP: $$v^s(S) = \frac{1}{Z}\sum_{w \in S} (z(w)+c)\mathbf{e}^w(w), \qquad (15)$$

where
$$Z = \sum_{w \in \mathrm{B} \cup \mathrm{M} \cup \mathrm{E} \cup \mathrm{S}} z(w) + c.$$

Dataset	MP	WP	SWP
NoteNotes	0.7257	0.7554	0.7544
MSRA	0.9276	0.9350	0.9349
Weibo	0.5772	**0.6124**	0.5702
Resume	0.9533	**0.9559**	0.9427
Average	0.7560	0.8131	0.8006

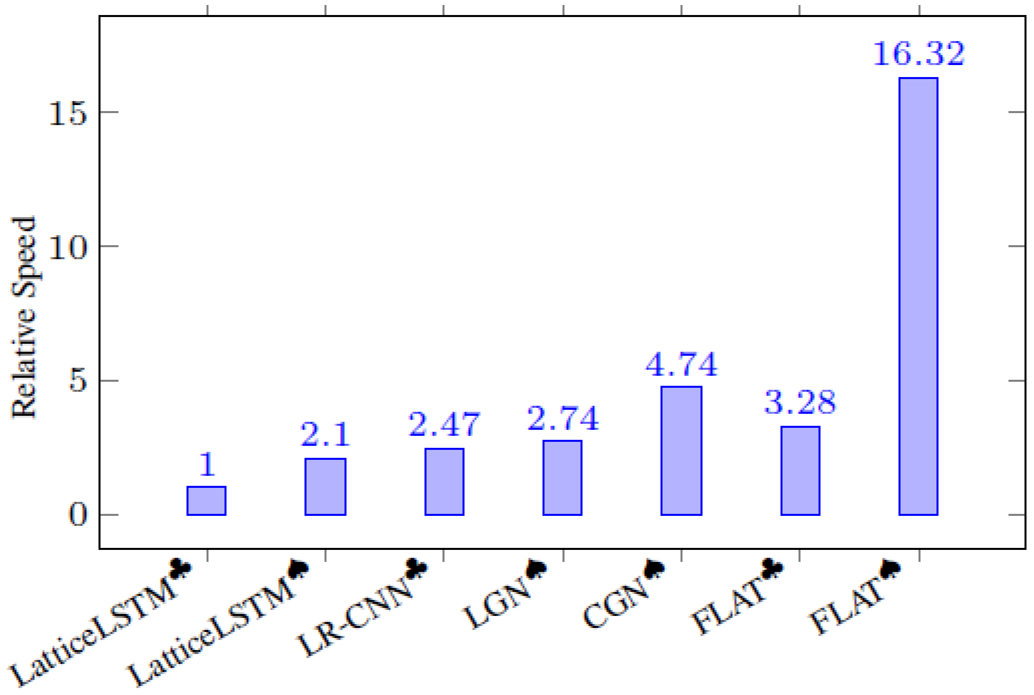
華東師範大學
EAST CHINA NORMAL UNIVERSITY

	Lexicon	Ontonotes	MSRA	Resume	Weibo
BiLSTM	-	71.81	91.87	94.41	56.75
TENER	-	72.82	93.01	95.25	58.39
Lattice LSTM	YJ	73.88	93.18	94.46	58.79
CNNR	YJ	74.45	93.71	95.11	59.92
LGN	YJ	74.85	93.63	95.41	60.15
PLT	YJ	74.60	93.26	95.40	59.92
FLAT	YJ	**76.45**	**94.12**	**95.45**	**60.32**
FLAT_{msm}	YJ	73.39	93.11	95.03	57.98
FLAT_{mld}	YJ	75.35	93.83	95.28	59.63
CGN	LS	74.79	93.47	94.12*	63.09
FLAT	LS	**75.70**	**94.35**	**94.93**	**63.42**

proposed (LSTM)	75.54	93.50	61.24	95.59

Models	**OntoNotes**	**MSRA**	**Weibo**	**Resume**
LR-CNN	74.45	93.71	59.92	95.11

華東師範大學
EAST CHINA NORMAL UNIVERSITY



Models	OntoNotes	MSRA	Weibo	Resume
Lattice-LSTM	11.99	14.78	11.09	15.61
LR-CNN	26.73	23.20	26.73	22.48
proposed (LSTM)	73.72	85.43	67.67	95.76
proposed (CNN)	**80.75**	**92.76**	**74.24**	**106.55**
proposed (Transformer)	63.86	58.18	58.75	61.11