Neural Word Segmentation Learning for Chinese

Zhen Cheng

Chinese Word Segmentation

- Input: x

  sentence/list of characters
- Output: y*

  list of words
- Definition:

$$y^* = \arg\max_{y \in \mathrm{GEN}(x)} \left(\sum_{i=1}^{n} \mathrm{score}(y_i | y_1, \cdots, y_{i-1}) \right)$$

- Challenges:
  - Ambiguity
  - Out-of-vocabulary words

Chinese Word Segmentation

Previous approaches:

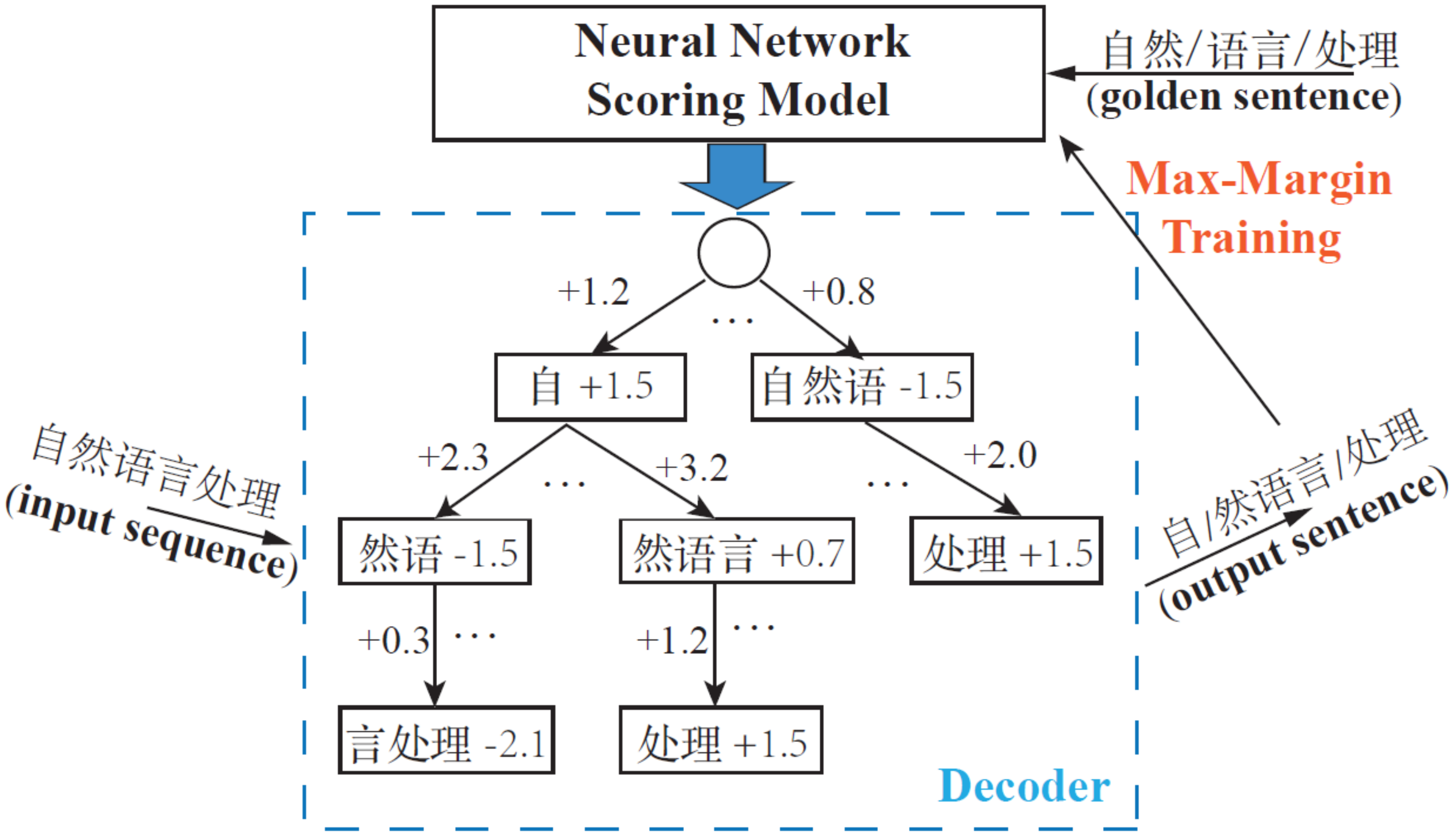
- Formalized as a sequence labeling task:
  - Character-based
  - Fixed size local windows

  Disadvantages: only contextual information, simple interactions

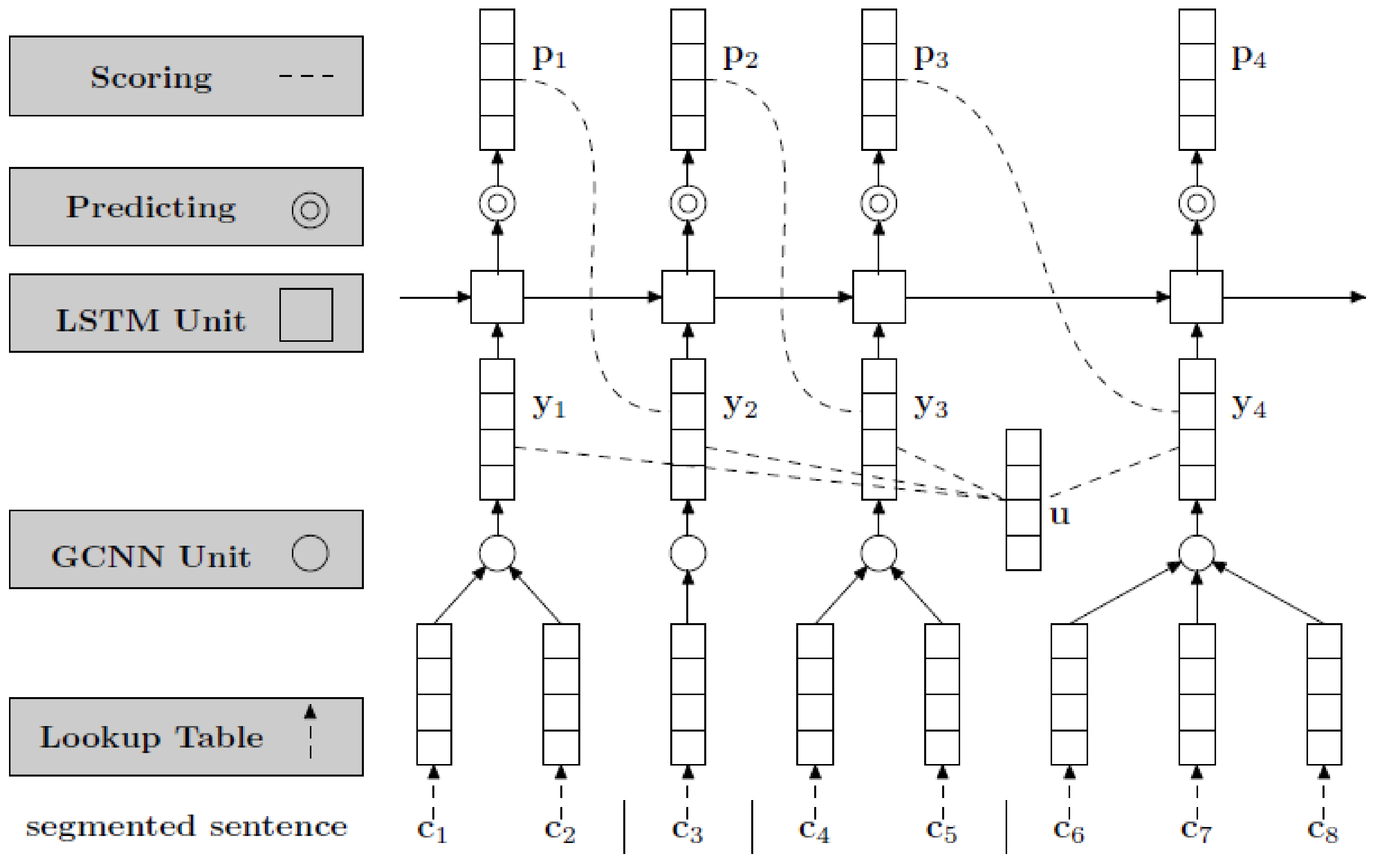
- Traditional methods:
  - Maximum Entropy(Berger et al., 1996)
  - Conditional Random Fields (Lafferty et al., 2001)

  Disadvantage: depends on the choice of features

GCNN-LSTM Model



GCNN-LSTM Model



GCNN-LSTM Model

- Features: directly evaluates the whole segmented sentences
    - Character Embedding
    - Gated Combination Neural Network
    - Sentence Score = Word Score + Link Score
    - Decoding with Beam Search
- Dataset:
    - PKU
    - MSR

Character Embedding

- Why not word embedding?
  - The data sparsity of n-gram
  - Words is derived from characters
  - Chain word candidates incrementally

counts	I	like	enjoy	deep	learning	NLP	modeling	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
modeling	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Gated Combination Neural Network

- Objective:

  To obtain word representation through characters

- Input:

  $\mathbf{c}_i \ (1 \leq i \leq L)$ d-dimensional character vector

- Output:

  $$\mathbf{w} = g(\mathbf{W}^{(L)} \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_L \end{bmatrix})$$ d-dimensional word vector

Gated Combination Neural Network

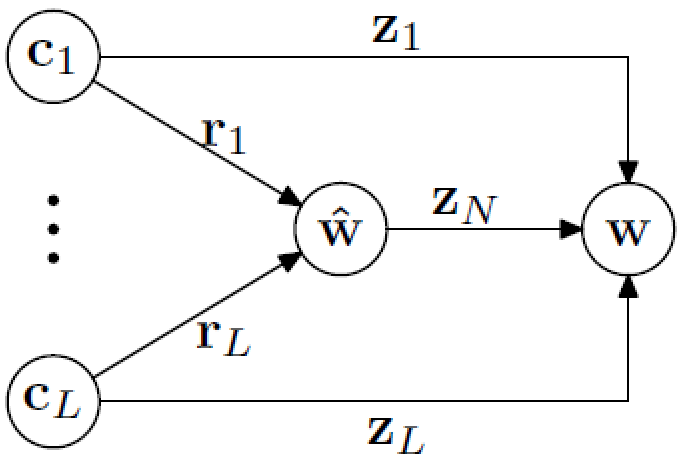
GCNN updates output function

- Word vector:

$$\mathbf{w} = \mathbf{z}_N \odot \hat{\mathbf{w}} + \sum_{i=1}^{L} \mathbf{z}_i \odot \mathbf{c}_i$$

- Update gates: $\mathbf{z}_N + \sum_{i=1}^{L} \mathbf{z}_i = \mathbf{1}$

$$\begin{bmatrix} \mathbf{z}_N \\ \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_L \end{bmatrix} = \exp(\mathbf{U}^{(L)} \begin{bmatrix} \hat{\mathbf{w}} \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_L \end{bmatrix}) \odot \begin{bmatrix} 1/\mathbf{Z} \\ 1/\mathbf{Z} \\ \vdots \\ 1/\mathbf{Z} \end{bmatrix}$$



Gated Combination Neural Network

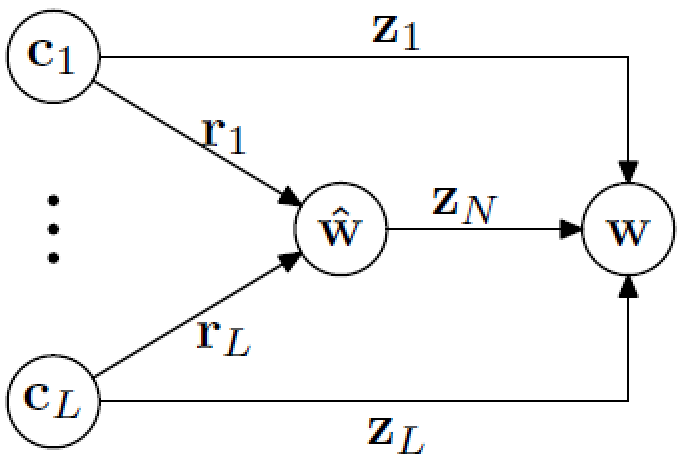
GCNN updates output function

- Activation:

$$\hat{\mathbf{w}} = \tanh(\mathbf{W}^{(L)} \begin{bmatrix} \mathbf{r}_1 \odot \mathbf{c}_1 \\ \vdots \\ \mathbf{r}_L \odot \mathbf{c}_L \end{bmatrix})$$

- Reset gates:

$$\begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_L \end{bmatrix} = \sigma(\mathbf{R}^{(L)} \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_L \end{bmatrix})$$



Sentence Score = Word Score + Link Score

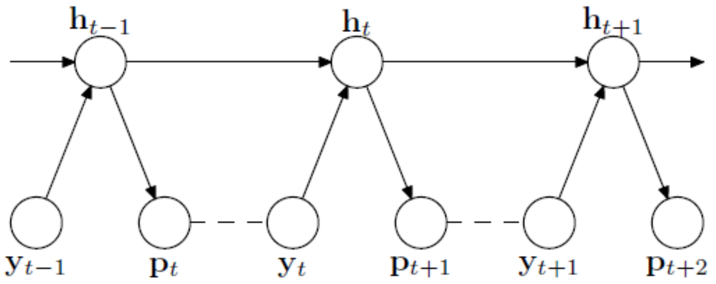
- Word Score: the score of the ith word

  $$\mathrm{Word\_Score}(\mathbf{y}_i) = \mathbf{u} \cdot \mathbf{y}_i$$

- Link Score: base on LSTM
  - A prediction p(t+1) about next word y(t+1):

    $$\mathbf{p}_{t+1} = \tanh(\mathbf{W}^p \mathbf{h}_t + \mathbf{b}^p)$$

    h(t) is the hidden state at time t
  - $\mathrm{Link\_Score}(\mathbf{y}_{t+1}) = \mathbf{p}_{t+1} \cdot \mathbf{y}_{t+1}$



- Sentence Score:

  $$s(y_{[1:n]}, \theta) = \sum_{t=1}^{n} (\mathbf{u} \cdot \mathbf{y}_t + \mathbf{p}_t \cdot \mathbf{y}_t)$$

Decoding with Beam Search

- Why not Viterbi search?
  - Viterbi uses Markov assumption
  - Want to capture the whole history of segmentation.

Decoding with Beam Search

Three steps for every position i

- Using GCNN to generate word vectors
- Generate segmentation ending at I
- Choose the k-max of candidates

Algorithm 1 Beam Search.

Input: model parameters θ
beam size k
maximum word length w
input character sequence $c[1:n]$

Output: Approx. k best segmentations

$\pi[0] \leftarrow \{(score = 0, \mathbf{h} = \mathbf{h}_0, \mathbf{c} = \mathbf{c}_0)\}$
for $i = 1$ to n **do**
  $\triangleright$ Generate Candidate Word Vectors
  $X \leftarrow \emptyset$
  for $j = \max(1, i - w)$ to i **do**
    $\mathbf{w} = \text{GCNN-Procedure}(c[j:i])$
    $X.\text{add}((index = j - 1, word = \mathbf{w}))$
  end for
  $\triangleright$ Join Segmentation
  $Y \leftarrow \{\ y.\text{append}(x) \mid y \in \pi[x.index]$ and $x \in X\}$
  $\triangleright$ Filter k-Max
  $\pi[i] \leftarrow k\text{-}\arg\max_{y \in Y} y.score$
end for
return $\pi[n]$

Training

- Structured margin loss:

$$\Delta(y^{(i)}, \hat{y}) = \sum_{t=1}^{m} \mu \mathbf{1}\{y^{(i),t} \neq \hat{y}^t\}$$

- Loss function:

$$J(\theta) = \frac{1}{|\Omega|} \sum_{(x^{(i)}, y^{(i)}) \in \Omega} l_i(\theta) + \frac{\lambda}{2} ||\theta||_2^2$$

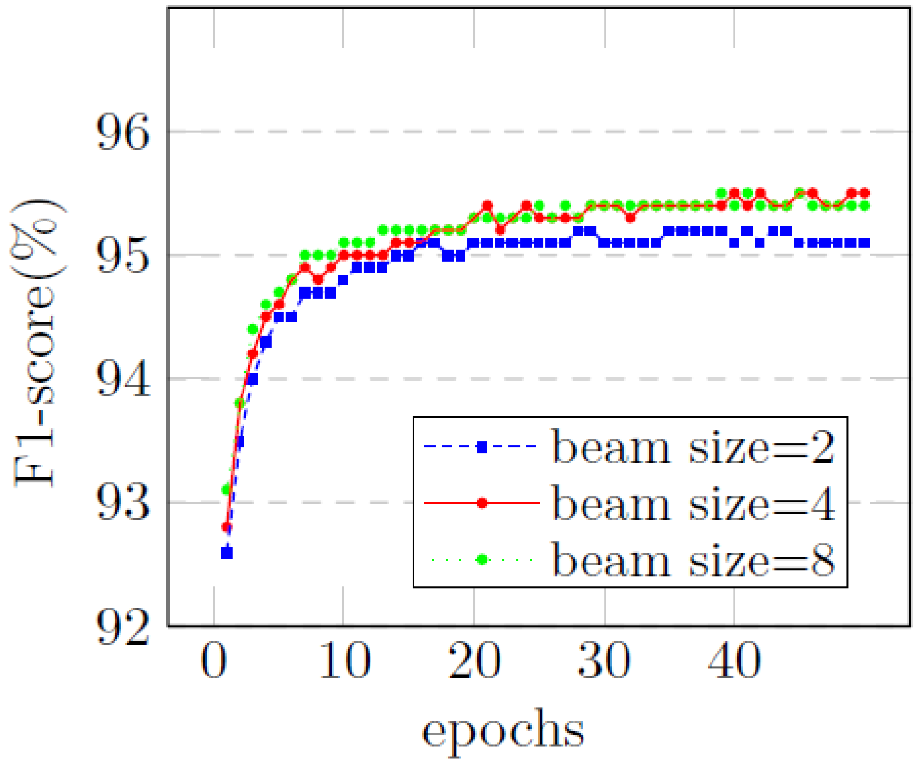
$$l_i(\theta) = \max_{\hat{y} \in \mathrm{GEN}(x^{(i)})} (s(\hat{y}, \theta) + \Delta(y^{(i)}, \hat{y}) - s(y^{(i)}, \theta))$$

- Parameter update:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i}$$

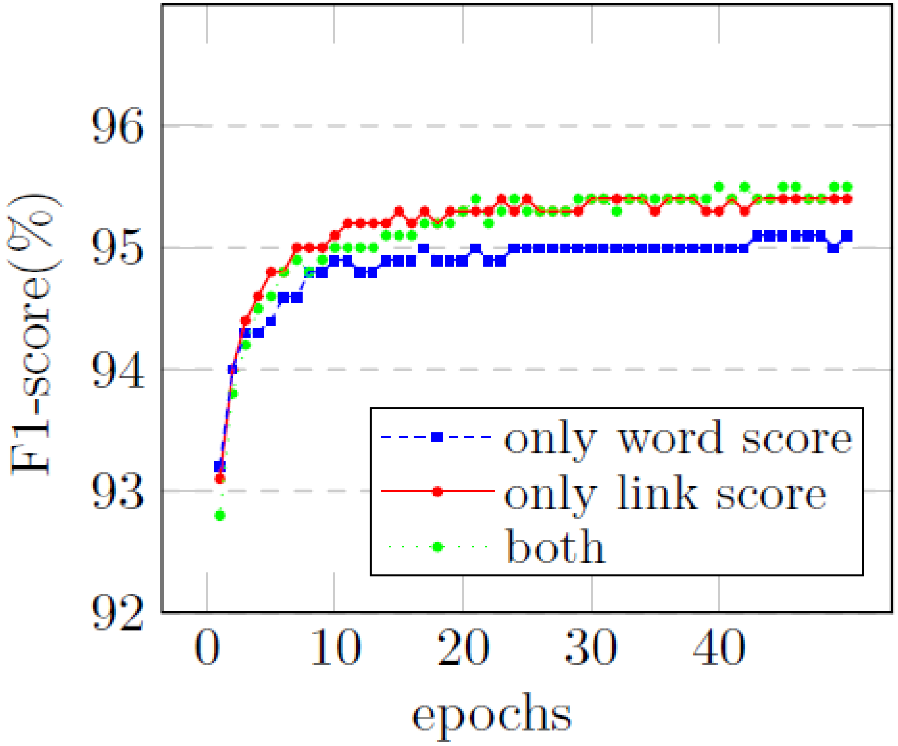
Performances of different beam sizes on PKU dataset

- Beam size = 4 is enough



Performances of different score strategies on PKU dataset

- Link score make sense because of the complete segmentation history



Results on MSR dataset with different maximum decoding word length settings

- Allowing longer words can improve the performance
- Time consuming should be considered

Max. word length	F_1 score	Time (Days)
4	96.5	4
5	96.7	5
6	96.8	6

Compare with other models

Models	PKU			MSR		
	P	R	F	P	R	F
(Zheng et al., 2013)	92.8	92.0	92.4	92.9	93.6	93.3
(Pei et al., 2014)	93.7	93.4	93.5	94.6	94.2	94.4
(Chen et al., 2015a)*	94.6	94.2	94.4	94.6	95.6	95.1
(Chen et al., 2015b) *	94.6	94.0	94.3	94.5	95.5	95.0
This work	**95.5**	**94.9**	**95.2**	**96.1**	**96.7**	**96.4**
+Pre-trained character embedding						
(Zheng et al., 2013)	93.5	92.2	92.8	94.2	93.7	93.9
(Pei et al., 2014)	94.4	93.6	94.0	95.2	94.6	94.9
(Chen et al., 2015a)*	94.8	94.1	94.5	94.9	95.9	95.4
(Chen et al., 2015b)*	95.1	94.4	94.8	95.1	96.2	95.6
This work	**95.8**	**95.2**	**95.5**	**96.3**	**96.8**	**96.5**

Compare with other models

Models	PKU	MSR	PKU	MSR
(Tseng et al., 2005)	95.0	96.4	-	-
(Zhang and Clark, 2007)	94.5	97.2	-	-
(Zhao and Kit, 2008b)	95.4	97.6	-	-
(Sun et al., 2009)	95.2	97.3	-	-
(Sun et al., 2012)	95.4	97.4	-	-
(Zhang et al., 2013)	-	-	96.1*	97.4*
(Chen et al., 2015a)	94.5	95.4	96.4*	97.6*
(Chen et al., 2015b)	94.8	95.6	96.5*	97.4*
This work	95.5	96.5	-	-