Subword Encoding in Lattice LSTM for Chinese Word Segmentation

Jie Yang¹, Yue Zhang², Shuailong Liang³
1. Harvard Medical School & Brigham and Women’s Hospital, Harvard University
2. Westlake University
3. Singapore University of Technology and Design

Overview

Chinese Word Segmentation:
- **Task**: converts Chinese character sequence to word sequence.
- **Features**: character + word + subword

Subwords Example:
- Matched Subwords in character sequence

Model/Experiments

Model Design:
- Take Bi-LSTM-CRF as the framework.
- Encodes the character sequence and matched subwords to predict the segmentation labels.
- Replace LSTM as Lattice LSTM encoder.
- Use Byte-Pair Encoding (BPE) to build the subword lexicon in an unsupervised way.

Lattice LSTM Framework:
- Extend the standard LSTM to enable multiple subword paths.
- Assign one gate to each subword path to control its weight.
- Weighted sum all the subword paths as the character mem cell.

Lattice LSTM Calculation:

```
\begin{align*}
& i_j = \sigma \left( \sum_{\ell \in \ell_j} \left( W_{i\ell} x_{\ell} + b_i \right) \right) \\
& f_j = \sigma \left( \sum_{\ell \in \ell_j} \left( W_{f\ell} x_{\ell} + b_f \right) \right) \\
& o_j = \sigma \left( \sum_{\ell \in \ell_j} \left( W_{o\ell} x_{\ell} + b_o \right) \right) \\
& c_j = \sum_{\ell \in \ell_j} \left( W_{c\ell} x_{\ell} + b_c \right) \\
& h_j = o_j \odot \tanh(c_j) \\
& \alpha_{j\ell} = \frac{\exp(i_j)}{\sum_{\ell' \in \ell_j} \exp(i_{\ell'})} \\
& \alpha_{j\ell} = \frac{\exp(f_j)}{\sum_{\ell' \in \ell_j} \exp(f_{\ell'})} \\
& \alpha_j = \sum_{\ell \in \ell_j} \alpha_{j\ell} \\
& \beta_j = \sum_{i \in \ell_j} \alpha_{i\ell} \\
& \gamma_j = \exp(h_j) \odot \sum_{\ell \in \ell_j} \alpha_{j\ell} \exp(h_{\ell})
\end{align*}
```

Experiments:
- **Baseline**: Bi-directional LSTM-CRF
- **Datasets**: CTB6/MSR/PKU/Weibo
- **Features**: Char unigram/bigram/subwords

Results:
- Significant improvement on all datasets compared with baseline.
- Comparable with SOTA models even with the model combining external labeled data.

Analysis

Sentence Length Performance
- **Baseline**: deep performance valley around 30-character sentences.
- **Subword based Lattice LSTM** gives better and more stable performance along all sentence length.

Who contributes the most?
- **Lexicon**: 4.5% error reduction
- **Pretrain emb**: 6.9-4.5%=2.4% error reduction
- **Future work**: use domain lexicon to improve the performance

Table 1: CWS frameworks

<table>
<thead>
<tr>
<th>Features</th>
<th>Character</th>
<th>Word</th>
<th>Subword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural</td>
<td>Chen, 2015</td>
<td>Zhang, 2016</td>
<td>This work</td>
</tr>
</tbody>
</table>

Table 2: Model performance

<table>
<thead>
<tr>
<th>Models</th>
<th>CTB6</th>
<th>SIGHAN</th>
<th>MSR</th>
<th>PKU</th>
<th>Weibo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zheng et al. (2013)</td>
<td>93.3</td>
<td>92.4</td>
<td>93.2</td>
<td>92.4</td>
<td>93.1</td>
</tr>
<tr>
<td>Pei et al. (2014)</td>
<td>97.2</td>
<td>93.2</td>
<td>96.6</td>
<td>95.1</td>
<td>97.2</td>
</tr>
<tr>
<td>Ma and Hinrichs (2015)</td>
<td>95.7</td>
<td>97.6</td>
<td>95.7</td>
<td>97.2</td>
<td>98.6</td>
</tr>
<tr>
<td>Liu et al. (2016)</td>
<td>96.0</td>
<td>97.7</td>
<td>95.7</td>
<td>97.2</td>
<td>98.1</td>
</tr>
<tr>
<td>Zhang et al. (2016)</td>
<td>95.8</td>
<td>96.3</td>
<td>96.1</td>
<td>97.1</td>
<td>98.6</td>
</tr>
<tr>
<td>Xu and Sun (2016)</td>
<td>96.5</td>
<td>97.6</td>
<td>95.7</td>
<td>97.1</td>
<td>98.6</td>
</tr>
<tr>
<td>Cai et al. (2017)</td>
<td>95.6</td>
<td>97.6</td>
<td>95.7</td>
<td>97.1</td>
<td>98.6</td>
</tr>
<tr>
<td>Chen et al. (2017)</td>
<td>95.6</td>
<td>97.6</td>
<td>95.7</td>
<td>97.1</td>
<td>98.6</td>
</tr>
<tr>
<td>Yang et al. (2017)</td>
<td>94.5</td>
<td>96.8</td>
<td>95.0</td>
<td>94.5</td>
<td>97.1</td>
</tr>
<tr>
<td>Ma et al. (2018)</td>
<td>96.7</td>
<td>97.4</td>
<td>96.1</td>
<td>97.1</td>
<td>98.6</td>
</tr>
<tr>
<td>Baseline</td>
<td>95.8</td>
<td>97.4</td>
<td>95.3</td>
<td>95.0</td>
<td>95.3</td>
</tr>
<tr>
<td>Lattice+Subword</td>
<td>96.1</td>
<td>97.8</td>
<td>95.8</td>
<td>95.3</td>
<td>95.3</td>
</tr>
</tbody>
</table>

Table 3: Lexicon vs Pretrain embeddings (CTB)

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>ERR%</th>
<th>R1/F1</th>
<th>R1/ERR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>95.9</td>
<td>95.6</td>
<td>95.8</td>
<td>0.0</td>
<td>96.7</td>
<td>77.3</td>
</tr>
<tr>
<td>Random Emb</td>
<td>96.1</td>
<td>95.8</td>
<td>95.0</td>
<td>-5.4</td>
<td>98.9</td>
<td>73.7</td>
</tr>
<tr>
<td>Pretrain Emb</td>
<td>96.2</td>
<td>95.9</td>
<td>96.0</td>
<td>-6.9</td>
<td>98.6</td>
<td>79.7</td>
</tr>
</tbody>
</table>

Case Study

- **Baseline error**: “多样性日”
- **Subwords**: “生物多样性”，“多样性”
- **Subwords can help the LatticeLSTM to split “多样性日” as “多样性” + “日”.

Figure 1: CWS task example

Figure 2: Characters/Subwords/Segmentation

Figure 3: Subword encoding for Chinese word segmentation with Lattice LSTM framework. Only forward Lattice LSTM is shown here, our model takes bidirectional Lattice LSTM.

Figure 4: F1-value against the sentence length.

Figure 5: Case Study in CTB