A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging

Wenbin Jiang
Institute of Computing Technology

Liang Huang
University of Pennsylvania

Qun Liu & Yajuan Lü
Institute of Computing Technology

ACL 2008 talk, Columbus, OH, June 2008.
prepared and presented by L. H.
Chinese Word Segmentation

民主
min-zhu
people-dominate

“democracy”
Chinese Word Segmentation

民主
min-zhu
people-dominate

江泽民 主席
jiang-ze-min zhu-xi
... - ... - people dominate - podium

“democracy”

“President Jiang Zemin”
Joint Chinese Segmentation and POS Tagging

Chinese Word Segmentation

民主
min-zhu
people-dominate

江泽民
jiang-ze-min

主席
zhu-xi

... - ... - people
dominate-podium

“democracy”

“President Jiang Zemin”
Chinese Word Segmentation

"people-dominate"

"democracy"

"Presidential Jiang Zemin"

This was 5 years ago.

Now Google is good at segmentation!
Word segmentation is needed for Chinese processing.

- First step in pipeline: its quality affects later modules.
- Especially in MT: mis-segmentation affects rule extraction.
- A lot of ambiguities:
  - "democracy"

- Sometimes unsolvable:
  - "President Jiang Zemin"

This work: **joint** word segmentation and POS tagging.

Now, Google is good at segmentation!
Chinese Word Segmentation

- word segmentation is needed for Chinese processing
  - first step in pipeline: its quality affects later modules
  - esp. in MT: mis-segmentation affects rule extraction
  - a lot of ambiguities:
  - sometimes unsolvable:
  - this work: joint word segmentation and POS tagging
Chinese Word Segmentation

• word segmentation is needed for Chinese processing
  • first step in pipeline: its quality affects later modules
  • esp. in MT: mis-segmentation affects rule extraction
  • a lot of ambiguities:
    - 下雨 天 地 面 积 水
    - 洽 谈 会 很 成 功 vs. 洽 谈 会 很 成 功
  • sometimes unsolvable:

• this work: joint word segmentation and POS tagging
Joint Chinese Segmentation and POS Tagging

- Chinese Word Segmentation
  - word segmentation is needed for Chinese processing
  - first step in pipeline: its quality affects later modules
  - esp. in MT: mis-segmentation affects rule extraction
  - a lot of ambiguities:
  - sometimes unsolvable:
  - this work: **joint** word segmentation and POS tagging
Overall Architecture

- cascaded approach (two levels of log-linear models)
  - core: perceptron with sparse features, word-level DP
  - outside-layer: extra probabilistic features, MERT

\[
g_1 = \sum_i \alpha_i \times f_i
\]

Word LM: \( g_2 = P_{wlm}(W) \)
POS LM: \( g_3 = P_{tlm}(T) \)
Labelling: \( g_4 = P(T|W) \)
Generating: \( g_5 = P(W|T) \)
Length: \( g_6 = |W| \)
Outline

- Dynamic Programming for Joint Segmentation/Tagging
- Core Features and the Perceptron
- Incorporating Non-Local Probabilistic Features
- Experiments
  - SIGHAN Bakeoff -- segmentation only
  - Chinese Treebank -- segmentation and tagging
Simple Dynamic Programming

- search through the lattice for the best path
- at each character
  - look back 1, 2, ... chars to make a potential word
  - score of this word = sum of features on each char
    - each char within the words is annotated \{b, m, e, s\}
    - each word is annotated with a POS tag
- \(O(n m T)\) - \(n\) sent. len.; \(m\) max word len.; \(T\) tagset

下雨天地面积 水 问题
Simple Dynamic Programming

- search through the lattice for the best path
- at each character
  - look back 1, 2, ... chars to make a potential word
  - score of this word = sum of features on each char
    - each char within the words is annotated \{b, m, e, s\}
    - each word is annotated with a POS tag
- $O(nmT) - n$ sent. len.; $m$ max word len.; $T$ tagset
Simple Dynamic Programming

- search through the lattice for the best path
- at each character
  - look back 1, 2, ... chars to make a potential word
  - score of this word = sum of features on each char
    - each char within the words is annotated \{b, m, e, s\}
    - each word is annotated with a POS tag
- $O(nmT) - n$ sent. len.; $m$ max word len.; $T$ tagset
Simple Dynamic Programming

- search through the lattice for the best path
- at each character
  - look back 1, 2, ... chars to make a potential word
  - score of this word = sum of features on each char
    - each char within the words is annotated \{b, m, e, s\}
    - each word is annotated with a POS tag
- \( O(nmT) \) - \( n \) sent. len.; \( m \) max word len.; \( T \) tagset
Simple Dynamic Programming

- search through the lattice for the best path
- at each character
  - look back 1, 2, ... chars to make a potential word
  - score of this word = sum of features on each char
    - each char within the words is annotated \{b, m, e, s\}
    - each word is annotated with a POS tag
- \(O(nmT)\) - \(n\) sent. len.; \(m\) max word len.; \(T\) tagset
Simple Dynamic Programming

- search through the lattice for the best path
- at each character
  - look back 1, 2, ... chars to make a potential word
  - score of this word = sum of features on each char
    - each char within the words is annotated \{b, m, e, s\}
    - each word is annotated with a POS tag
- \(O(nmT)\) - \(n\) sent. len.; \(m\) max word len.; \(T\) tagset
Simple Dynamic Programming

- search through the lattice for the best path
- at each character
  - look back 1, 2, ... chars to make a potential word
  - score of this word = sum of features on each char
    - each char within the words is annotated \{b, m, e, s\}
    - each word is annotated with a POS tag
- \(O(nmT)\) - \(n\) sent. len.; \(m\) max word len.; \(T\) tagset

\[
\begin{array}{cccc}
\text{b} & \text{m} & \text{m} & \text{e} \\
\end{array}
\]
Simple Dynamic Programming

- search through the lattice for the best path
- at each character
  - look back 1, 2, ... chars to make a potential word
  - score of this word = sum of features on each char
    - each char within the words is annotated \{b, m, e, s\}
    - each word is annotated with a POS tag
- $O(nmT)$ - $n$ sent. len.; $m$ max word len.; $T$ tagset

- 下 雨 天 地 面 积 水 问 题
  - 下雨天 地面积 水 问题

Joint Chinese Segmentation and POS Tagging
Feature Templates

- word-level dynamic programming, char-level features
- each character is annotated by \{b, m, e, s\} scheme
  - plus the POS tag
- characters around the current char (local window)
- is the current character a punctuation? (Ng & Low 04)
Feature Templates

- word-level dynamic programming, char-level features
- each character is annotated by \{b, m, e, s\} scheme
- plus the POS tag
- characters around the current char (local window)
- is the current character a punctuation? (Ng & Low 04)

b e

下雨天 地 面 积 水 问 题
Perceptron

• **online algorithm:** several passes over the training data
  • in each iteration, decode each sentence
  • compare with gold-standard, update weights if needed

• averaging helps counter over-fitting

2: \( w \leftarrow 0 \)  
3: \textbf{for} \( t \leftarrow 1 \ldots T \) \textbf{do}  
4: \textbf{for} \( i \leftarrow 1 \ldots N \) \textbf{do}  
5: \( \hat{y} = \arg\max_{y \in \text{cand}(s_i)} w \cdot f(y) \)  
6: \textbf{if} \( \hat{y} \neq y_i^+ \) \textbf{then}  
7: \( w \leftarrow w + f(y_i^+) - f(\hat{y}) \)  
8: \textbf{return} \( w \)
Perceptron

- **online algorithm:** several passes over the training data
  - in each iteration, decode each sentence
  - compare with gold-standard, update weights if needed
- averaging helps counter over-fitting

2: \( w \leftarrow 0 \)
3: for \( t \leftarrow 1 \ldots T \) do
4: \hspace{1em} for \( i \leftarrow 1 \ldots N \) do
5: \hspace{2em} \hat{y} = \operatorname{argmax}_{y \in \text{cand}(s_i)} w \cdot f(y) \)
6: \hspace{2em} if \( \hat{y} \neq y_i^+ \) then
7: \hspace{3em} w \leftarrow w + f(y_i^+) - f(\hat{y})
8: return \( w \)

▷ initial weights
▷ \( T \) iterations

feature representation
Outline

• Dynamic Programming for Joint Segmentation/Tagging
• Core Features and the Perceptron
• Incorporating Non-Local Probabilistic Features
• Experiments
  • SIGHAN Bakeoff -- segmentation only
  • Chinese Treebank -- segmentation and tagging
Adding Non-Local Features

- features in the perceptron are within a local window
- but we also need more context, say, POS tag seq.
- but non-local features are hard to incorporate!

- our idea: borrow from machine translation

\[ g_1 = \sum_i \alpha_i \times f_i \]

Core Linear Model (Perceptron)

\[ S = \sum_j w_j \times g_j \]

Outside-layer Linear Model

- Word LM: \( g_2 = P_{wlm}(W) \)
- POS LM: \( g_3 = P_{tlm}(T) \)
- Labelling: \( g_4 = P(T|W) \)
- Generating: \( g_5 = P(W|T) \)
- Length: \( g_6 = |W| \)
Adding Non-Local Features

- features in the perceptron are within a local window
- but we also need more context, say, POS tag seq.
- but non-local features are **hard** to incorporate!
  - our idea: borrow from machine translation

```
Core Linear Model (Perceptron)
\[ g_1 = \sum_i \alpha_i \times f_i \]

Outside-layer Linear Model
\[ S = \sum_j w_j \times g_j \]
```

probabilistic features rather than sparse features!
Extra Features

- $n$-gram sequence models
  - Word Trigram Model
  - POS 4-gram Model
- Word-POS co-occurrence statistics
  - $P(t \mid w)$ and $P(w \mid t)$
- Word Count Penalty (otherwise tend to have longer words)
- use minimum-error rate training (Och, 2003) to tune
  - core-perceptron as one feature (most of the weight)
  - $n$-best list from core-perceptron for extra features
Extra Features

- *n*-gram sequence models
  - Word Trigram Model
  - POS 4-gram Model

- Word-POS co-occurrence statistics
  - \( P(t \mid w) \) and \( P(w \mid t) \)

- Word Count Penalty (otherwise tend to have longer words)

- Use minimum-error rate training (Och, 2003) to tune
  - core-perceptron as one feature (most of the weight)

- *n*-best list from core-perceptron for extra features
Extra Features

- *n*-gram sequence models
  - Word Trigram Model
  - POS 4-gram Model
- Word-POS co-occurrence statistics
  - $P(t|w)$ and $P(w|t)$
- Word Count Penalty (otherwise tend to have longer words)
- use minimum-error rate training (Och, 2003) to tune
  - core-perceptron as one feature (most of the weight)
  - *n*-best list from core-perceptron for extra features
• *n*-gram sequence models
  • Word Trigram Model
  • POS 4-gram Model
• Word-POS co-occurrence statistics
  • \( P(t \mid w) \) and \( P(w \mid t) \)
• Word Count Penalty (otherwise tend to have longer words)
• use minimum-error rate training (Och, 2003) to tune
  • core-perceptron as one feature (most of the weight)
• *n*-best list from core-perceptron for extra features
Experiments

1. SIGHAN Bakeoff -- segmentation only
2. Chinese Treebank -- segmentation and tagging
Experiments - SIGHAN Bakeoff 2

- Compare against SIGHAN Bakeoff 2 best result
- closed test (no extra annotation used in training)
- report segmentation accuracy F-measure

<table>
<thead>
<tr>
<th>dataset</th>
<th>SIGHAN best</th>
<th>Zhang &amp; Clark 07</th>
<th>this work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Academia Sinica</td>
<td>0.952</td>
<td>0.946</td>
<td>0.954</td>
</tr>
<tr>
<td>CityU of HK</td>
<td>0.943</td>
<td>0.951</td>
<td>0.958</td>
</tr>
<tr>
<td>Peking Univ.</td>
<td>0.950</td>
<td>0.945</td>
<td>0.940</td>
</tr>
<tr>
<td>Microsoft Research</td>
<td>0.964</td>
<td>0.972</td>
<td>0.975</td>
</tr>
</tbody>
</table>
Experiments - CTB 5

- test Joint Segmentation and Tagging on CTB 5
  - standard split (18k sent. in training; ~350 sent. for dev/test)
- Core Perceptron only
- Ng & Low 04 on CTB 3 (different split; not reproducible)

<table>
<thead>
<tr>
<th>Training</th>
<th>Test Task</th>
<th>Core Model</th>
<th>Ng/Low 04 on CTB3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segmentation</td>
<td>Segmentation</td>
<td>0.971</td>
<td>-</td>
</tr>
<tr>
<td>Joint S &amp; T</td>
<td>Segmentation</td>
<td>0.973</td>
<td>0.952</td>
</tr>
<tr>
<td>Joint S &amp; T</td>
<td>Joint S &amp; T</td>
<td>0.925</td>
<td>0.919</td>
</tr>
</tbody>
</table>

cross validation split
Contribution of Extra Features

- adding all extra features help a lot
  - Segment.: 97.3 => 97.85; Joint S & T: 92.5 => 93.41
- but effect of excluding individual feature is negligible
Contribution of Extra Features

- adding all extra features help a lot
  - Segment.: 97.3 => 97.85; Joint S & T: 92.5 => 93.41
- but effect of excluding individual feature is negligible
Contribution of Extra Features

- adding all extra features help a lot
  - Segment.: 97.3 => 97.85; Joint S & T: 92.5 => 93.41
- but effect of excluding individual feature is negligible

### Graph

<table>
<thead>
<tr>
<th>Feature</th>
<th>Segmentation</th>
<th>Joint S &amp; T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>All</td>
<td>95</td>
<td>95</td>
</tr>
<tr>
<td>-Core</td>
<td>91</td>
<td>91</td>
</tr>
<tr>
<td>-wLM</td>
<td>87</td>
<td>87</td>
</tr>
<tr>
<td>-POSLM</td>
<td>83</td>
<td>83</td>
</tr>
<tr>
<td>-p(w</td>
<td>t)</td>
<td></td>
</tr>
<tr>
<td>-p(t</td>
<td>w)</td>
<td></td>
</tr>
<tr>
<td>-LenPen</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*most significant*
Conclusion

- cascaded model for joint segmentation and tagging
  - core-level: perceptron with sparse features
  - outside-layer: extra non-local prob. features (mert)
- simple architecture; fast in practice
- state-of-the-art performance in both segmentation (SIGHAN) and joint segmentation and tagging (CTB)
- future work
  - incorporating more sophisticated non-local features
  - semi-supervised learning (CTB is too small)
Thank you!

谢谢 大家

VV  NN