IWSLT – Trento, Italy – October 2007

Improved Chunk-level Reordering for Statistical Machine Translation

Yuqi Zhang, Richard Zens and Hermann Ney

Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik 6
Computer Science Department
RWTH Aachen University, Germany
Overview

- Introduction
- Phrase-based SMT
- Chunk-level reordering system
- Improvements
  - Reordering training data
  - Reordering-lattice weighting
- Results
- Conclusion and Outlook
Introduction

goal: improve MT utilizing syntactic knowledge

idea: reordering at the chunk level

approach:
1. chunk source sentence
2. reorder chunks
3. represent alternative reorderings in a lattice
4. translate lattice
Phrase-based SMT

log-linear model:

\[ P_r(e_1^I | f_1^J) = \frac{\exp \left( \sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J) \right)}{\sum_{I', e_1'^I} \exp \left( \sum_{m=1}^{M} \lambda_m h_m(e_1'^I, f_1^J) \right)} \]

models:

- phrase translation model
- phrase count features
- word-based translation model
- word and phrase penalty
- target language model (6-gram)
- distortion model
System Structure

Standard Translation Process

Source Text Sentences

SMT System

Translation Output

Translation Process with Source Reordering

Source Text Sentences

POS Tagging

Shallow Chunking

Reordering Rules

Source Reordering Lattice

SMT System

Translation Output
An Example

<table>
<thead>
<tr>
<th>source</th>
<th>可以 但 是 我 们 出 租 车 不 多</th>
</tr>
</thead>
<tbody>
<tr>
<td>pin yin</td>
<td>ke yi dan shi wo men chu zu che bu duo</td>
</tr>
<tr>
<td>POS</td>
<td>v    c    r   v   n   d   m</td>
</tr>
<tr>
<td>chunks</td>
<td>v    c    r   NP   VP</td>
</tr>
</tbody>
</table>

*English gloss*: yes but we taxi not many

**used reordering rules**

- NP VP → VP NP
- r NP VP → r VP NP
- r NP VP → VP r NP

**Reordering Lattice:**

![Reordering Lattice Diagram]
Reordering Rules Extraction

- convert word-to-word alignment to chunk-to-word alignment

- run standard phrase extraction on chunk-to-word alignment
Reordering Rules Extraction (cont’d)

- extract rules from monotone phrases and reordering phrases
  
  - e.g. $NP_0NP_1 \# NP_0 NP_1$ $NP_0NP_1 \# NP_1 NP_0$

(a) monotone phrase, (b) reordering phrase, (c) cross phrase
Reordering Lattice Generation I

- apply reordering rules to chunked source sentence
- represent alternative reorderings as a lattice
- example:

```
NP  NP  v
[ 上海 浦东 ] [ 开发 与 法制 建设 ] 并存
f0  f1  f2  f3  f4  f5  f6
NP  NP  #  0  1

NP  NP  #  1  0

NP  v  #  0  1

NP  v  #  1  0

NP  NP  v  #  0  1  2

NP  NP  v  #  1  2  0

NP  NP  v  #  2  0  1

Sentence Permutations
0  1  2  3  4  5  6
2  3  4  5  0  1  6
0  1  2  3  4  5  6
0  1  6  2  3  4  5
0  1  2  3  4  5  6
2  3  4  5  6  0  1
6  0  1  2  3  4  5
```
Reordering Lattice Generation II

• chunk-level lattice:

• word-level lattice:
Training Data Reordering I

- chunk source training data
- generate chunk-to-word alignment
- reorder source chunks to monotonize alignments.
- train LM on reordered source training data
- extract phrases on reordered training data
Training Data Reordering II

Source Sentence → Target sentence → Reordered Source Sentence

Training

Alignment 1 → Phrase Table 1

Alignment 2 → Phrase Table 2

Phrase Table 1+2
Lattice Weighting

- For each path in the lattice, the weight is computed by the two models
  - reordered source language model $h_{slm}$
  - reordering rules probability model $h_{reorder}$
Lattice Weighting: $h_{\text{slm}}$

- Each path of the lattice is a permutation $f_{\pi_1}^{\pi J} = f_{\pi_1}, \ldots, f_{\pi J}$ for a given source sentence $f_1^J$

\[
h_{\text{slm}}(f_{\pi_1}^{\pi J}, f_1^J) = \log p(f_{\pi_1}^{\pi J} | f_1^J)
\]

$\pi_j$ is the permutation position of word $f_j$

- Word trigram language model

\[
\log p(f_{\pi_1}^{\pi J} | f_1^J) = \sum_{j=1}^{J} \log p(f_{\pi_j} | f_{\pi_j-1}, f_{\pi_j-2})
\]
Lattice Weighting: $h_{\text{reorder}}$

\[
h_{\text{reorder}}(\pi_1^N, c_1^N) = \log(p(\pi_1^N|c_1^N))
\]

$c_1^N$: sequence of chunks, $f_1^J = c_1^N$

$\pi_n$: permutation position of chunk $c_n$.

\[
p(\pi_1^N|c_1^N) = \sum_B \alpha(c_1^N) \cdot p(\pi_1^N|c_1^N, B)
\]

\[
p(\pi_1^N|c_1^N, B) = p(\tilde{\pi}_1^K|\tilde{c}_1^K) = \prod_{k=1}^{K} p(\tilde{\pi}_k|\tilde{c}_k) = \prod_{k=1}^{K} \frac{N(\tilde{\pi}_k, \tilde{c}_k)}{N(\tilde{c}_k)}
\]

$B$: segmentation

$\tilde{c}_k$: left-hand side of $r_k$

$\tilde{\pi}_k$: right-hand side $r_k$
## Corpus Statistics

<table>
<thead>
<tr>
<th></th>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Train</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>43 k</td>
<td></td>
</tr>
<tr>
<td>Words</td>
<td>380 k</td>
<td>420 k</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>11 760</td>
<td>9 933</td>
</tr>
<tr>
<td><strong>Dev dev2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>Words</td>
<td>3 578</td>
<td>3 908</td>
</tr>
<tr>
<td>OOVs</td>
<td>73</td>
<td>–</td>
</tr>
<tr>
<td><strong>Test dev3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>506</td>
<td></td>
</tr>
<tr>
<td>Words</td>
<td>3 837</td>
<td>3 970</td>
</tr>
<tr>
<td>OOVs</td>
<td>70</td>
<td>–</td>
</tr>
</tbody>
</table>

- optimize on BLEU score.
Translation Result I

Translation performance for the Chinese-English IWSLT05 task

<table>
<thead>
<tr>
<th>test(dev3)</th>
<th>WER[%]</th>
<th>PER[%]</th>
<th>TER[%]</th>
<th>BLEU[%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline: chunk reorder</td>
<td>33.5</td>
<td>27.2</td>
<td>32.0</td>
<td>59.0</td>
</tr>
<tr>
<td>+ ruleProb</td>
<td>33.1</td>
<td>27.0</td>
<td>32.0</td>
<td>59.7</td>
</tr>
<tr>
<td>+ reordered train data</td>
<td>32.7</td>
<td>27.8</td>
<td>31.5</td>
<td>60.3</td>
</tr>
</tbody>
</table>

- baseline: reordering lattice is weighted by source language model.
## Comparison with the RWTH best system

<table>
<thead>
<tr>
<th></th>
<th>BLEU[%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>monotone</td>
<td>56.0</td>
</tr>
<tr>
<td>RWTH best system</td>
<td>62.4</td>
</tr>
<tr>
<td>source reorder improved</td>
<td>60.3</td>
</tr>
</tbody>
</table>
## Translation Examples

<table>
<thead>
<tr>
<th>source</th>
<th>有很多鱼的地方在哪？</th>
</tr>
</thead>
<tbody>
<tr>
<td>chunks</td>
<td>有_v [NP 很多_m 鱼_n] 的_u 地方_n 在_p [NP 哪_r] ?_w</td>
</tr>
<tr>
<td>reference</td>
<td>What place has a lot of fish?</td>
</tr>
<tr>
<td>chunk reorder</td>
<td>Where can i find a lot of fish?</td>
</tr>
<tr>
<td>RWTH-best-system</td>
<td>there are many fish Where?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>source</th>
<th>我想要一个面向海滩的房间.</th>
</tr>
</thead>
<tbody>
<tr>
<td>chunks</td>
<td>我_r 想_v 要_v 一个_m [VP 面向_v 海滩_n] 的_u 房间_n _w</td>
</tr>
<tr>
<td>reference</td>
<td>I’d like a room facing the beach.</td>
</tr>
<tr>
<td>chunk reorder</td>
<td>I would like a room facing the beach.</td>
</tr>
<tr>
<td>RWTH-best-system</td>
<td>I would like a beach facing the room.</td>
</tr>
</tbody>
</table>
Summary

• idea:
  1. chunk input sentence
  2. reorder chunks
  3. represent alternative reorderings as lattice
  4. translate lattice

• improve system
  1. reorder training data
  2. rule probability model to lattice
Outlook

- large data task (e.g. NIST)
- other language pairs
- improve chunk parsing
- analyze what kind of rules work well
THANK YOU FOR YOUR ATTENTION!
Chunk Parsing

- POS tagging + word segmentation with ICTCLAS tool (from Institute of Computing Technology, Chinese Academy of Sciences)

- Learn chunks from Chinese Treebank (LDC2005T01) with the constraints:
  - a subtree has more than one child,
  - the children of a subtree are all leaves.

- Tag each source word to mark what chunk it belongs to and its position within a chunk with Maximum Entropy Tagging (YASMET tool)
  - input features: word + POS tag
  - output: chunk types + chunk boundary
Chunking Result

Statistics of training and test corpus for chunk parsing (from Chinese Treebank LDC2005T01)

<table>
<thead>
<tr>
<th></th>
<th>train</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>sentences</td>
<td>17 785</td>
<td>1 000</td>
</tr>
<tr>
<td>words</td>
<td>486 468</td>
<td>21 851</td>
</tr>
<tr>
<td>chunks</td>
<td>105 773</td>
<td>4 680</td>
</tr>
<tr>
<td>words out of chunks</td>
<td>244 416</td>
<td>10 282</td>
</tr>
</tbody>
</table>

Result of tagging: found: 4414 chunks; correct:2879

<table>
<thead>
<tr>
<th>accuracy (%)</th>
<th>precision (%)</th>
<th>recall (%)</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>74.51</td>
<td>65.2</td>
<td>61.5</td>
<td>63.3</td>
</tr>
</tbody>
</table>

- The number of chunk types: 24
- a chunk is correct when both chunk type and boundary are correct
- precision and recall are at chunk level
- accuracy: correct tags at word level
## Rules Statistics

<table>
<thead>
<tr>
<th></th>
<th>rules</th>
<th>singletons(%)</th>
<th>reorder rules</th>
<th>used rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos rules</td>
<td>327k</td>
<td>287k (88%)</td>
<td>118k (36%)</td>
<td>49k</td>
</tr>
<tr>
<td>chunk rules</td>
<td>184k</td>
<td>162k (88%)</td>
<td>63k (34%)</td>
<td>25k</td>
</tr>
</tbody>
</table>
Detail Statistics of Chunk Rules

![Bar chart showing the distribution of chunk rules by length and type (total rules, singletons, reorder rules).]