Synchronous Binarization for Machine Translation

Hao Zhang (Rochester)  Liang Huang (Penn)
Dan Gildea (Rochester)  Kevin Knight (USC/ISI)

presented by Liang Huang
Outline

• Overview: Syntax-based Machine Translation
  • Synchronous Grammars
  • Translation as Parsing
  • ISI Syntax-based System
• The Problem of Binarization
• Our Solution: Synchronous Binarization
  • Linear-time Binarization Algorithm
• Experiments
Synchronous CFGs

• Synchronous Context-Free Grammar (SCFG)
• CFG in two dimensions, generating pairs of trees/strings
• co-indexed nonterminal further rewritten as a unit

S → NP(1) PP(2) VP(3), NP(1) VP(3) PP(2)
NP → Baoweier,
PP → yu Shalong,
VP → juxing le huitan,

Powell
with Sharon
held a meeting


HLT-NAACL 2006
Translation as Parsing

• Many problems in syntax-based MT reduce to parsing

• here we consider decoding as monolingual parsing

\[ S \rightarrow NP^{(1)} \ PP^{(2)} \ VP^{(3)} , \ NP^{(1)} \ VP^{(3)} \ PP^{(2)} \]

\[ NP \rightarrow \text{Baoweier,} \quad \text{NP}^{(1)} \text{ Powell} \]

\[ PP \rightarrow \text{yu Shalong,} \quad \text{PP}^{(2)} \text{ with Sharon} \]

\[ VP \rightarrow \text{juxing le huitan,} \quad \text{VP}^{(3)} \text{ held a meeting} \]

\[ \text{Baoweier yu Shalong juxing le huitan} \]

\[ \text{Powell held a meeting with Sharon} \]


HLT-NAACL 2006
Translation as Parsing

• Many problems in syntax-based MT reduce to parsing
  
• Here we consider decoding as monolingual parsing

\[
\begin{align*}
S & \rightarrow \text{NP}^{(1)} \text{ PP}^{(2)} \text{ VP}^{(3)}, \\
\text{NP} & \rightarrow \text{Baoweier,} \\
\text{PP} & \rightarrow \text{yu Shalong,} \\
\text{VP} & \rightarrow \text{juxing le huitan,}
\end{align*}
\]

---


HLT-NAACL 2006 Synchronous Binarization
Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
- here we consider decoding as monolingual parsing

\[
S \rightarrow NP^{(1)} PP^{(2)} VP^{(3)},
\]

NP → Baoweier,
PP → yu Shalong,
VP → juxing le huitan,

\[
\begin{array}{ccccccc}
NP & NP & NP & NP & NP & NP & NP \\
Baoweier & yu Shalong & juxing le huitan & Powell & & & \\
1 & 2 & 3 & 4 & 5 & 6 & 7
\end{array}
\]


HLT-NAACL 2006
Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
- Here we consider \textit{decoding} as monolingual parsing

\[
S \rightarrow NP^{(1)} \ PP^{(2)} \ VP^{(3)},
\]

\[
NP \rightarrow \text{Baoweier},
\]

\[
PP \rightarrow \text{yu Shalong},
\]

\[
VP \rightarrow \text{juxing le huitan},
\]
Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
- Here we consider **decoding** as monolingual parsing

\[
S \rightarrow \text{NP}(1) \text{ PP}(2) \text{ VP}(3), \\
\text{NP} \rightarrow \text{Baoweier}, \\
\text{PP} \rightarrow \text{yu Shalong}, \\
\text{VP} \rightarrow \text{juxing le huitan},
\]

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>PP</td>
<td>VP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baoweier</td>
<td>yu Shalong</td>
<td>juxing le huitan</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Powell</td>
<td>with Sharon</td>
<td>held a meeting</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


HLT-NAACL 2006

Synchronous Binarization
Translation as Parsing

• Many problems in syntax-based MT reduce to parsing

• Here we consider decoding as monolingual parsing

\[ S \rightarrow NP^{(1)} \, PP^{(2)} \, VP^{(3)}, \]

\[ NP \rightarrow Baoweier, \]

\[ PP \rightarrow yu Shalong, \]

\[ VP \rightarrow juxing le huitan, \]


HLT-NAACL 2006
Translation as Parsing

- Many problems in syntax-based MT reduce to parsing
- Here we consider **decoding** as monolingual parsing

\[
S \rightarrow \text{NP}^{(1)} \text{ PP}^{(2)} \text{ VP}^{(3)}, \\
\text{NP} \rightarrow \text{Baoweier}, \\
\text{PP} \rightarrow \text{yu Shalong}, \\
\text{VP} \rightarrow \text{juxing le huitan},
\]

Where are these rules from?

- Induce synchronous grammars from parallel corpora
- ISI syntax-based system

\[ S \rightarrow NP^{(1)} PP^{(2)} VP^{(3)}, \quad NP^{(1)} VP^{(3)} PP^{(2)} \]

\[ NP \rightarrow Baoweier, \quad Powell \]

\[ PP \rightarrow yu Shalong, \quad with Sharon \]

\[ VP \rightarrow juxing le huitan, \quad held a meeting \]

(Galley et al., 2004)
Efficient parsing requires binary-branching grammars
- e.g., CKY (explicit) or Earley (implicit)

Many ways to binarize
- They all work fine

\[ S \rightarrow \text{NP VP PP} \]

\[ S \rightarrow \text{NP PP VP} \]
Integrating Language Models

- dynamic programming algorithm
- remembering “boundary words”
- different binarization has very different effect

Powell
NP
with ... Sharon
PP
held ... meeting
VP

Baoweier yu Shalong juxing le huitan

(Wu, 1996)   (Chiang, 2005)
Integrating Language Models

- dynamic programming algorithm
- remembering “boundary words”
- different binarization has very different effect

Baoweier yu Shalong juxing le huitan

Powell NP
with PP
held VP

held VP-PP

Sharon

held VP

meeting

(Wu, 1996) (Chiang, 2005)
Integrating Language Models

- dynamic programming algorithm
- remembering “boundary words”
- different binarization has very different effect

Powell ... with ... Sharon
NP-PP

Baoweier yu Shalong juxing le huitan
Binarization in 2D

\[ S \rightarrow \]

NP VP PP
|
NP PP VP

\[ S \rightarrow \] NP VP PP
|
NP PP VP

Sharon
meeting
Powell

with a held

VP PP

Sharon with
meeting a
held

NP

PP

VP

Baoweier
Yu Shalong
Juxing
Le Huitan

Synchronous Binarization
Binarization in 2D

S → NP VP PP
   NP PP VP

Sharon
... with
meeting
... held
Powell

Baoweier  yu  Shalong  juxing  le  huitan

NP PP VP
Binarization in 2D

S →

\[
S → \frac{NP}{NP} \frac{VP}{VP} \frac{PP}{PP} \frac{VP}{VP}
\]

Sharon

... with meeting

... held

Powell

NP

PP

VP

Baoweier

yu

Shalong

juxing

le

huitan

Synchronou
Synchronous Binarization

- converts an SCFG into binary-branching
- intuition: should have contiguous spans on both sides
- binarizing the permutation of nonterminals

\[ S \rightarrow \begin{array}{ccc} NP & VP & PP \\ \times & & \\
NP & PP & VP \end{array} \]

(1,3,2)

1

(3,2)

3

2

Sharon...with
meeting
...held

PP

VP

NP

Baoweier yu Shalong juxing le huitan
Is every rule binarizable?
Is every rule binarizable?

- for $n < 4$, yes
Is every rule binarizable?

- for $n < 4$, yes
- for $n \geq 4$, many permutations non-binarizable
Is every rule binarizable?

- for \( n < 4 \), yes
- for \( n \geq 4 \), many permutations non-binarizable
- \( Q \): are these bad cases linguistically motivated?
Is every rule binarizable?

- for \( n < 4 \), yes
- for \( n \geq 4 \), many permutations non-binarizable
- Q: are these bad cases linguistically motivated?
  - Wu (1997) conjectured no
Is every rule binarizable?

• for $n < 4$, yes
• for $n \geq 4$, many permutations non-binarizable
• Q: are these bad cases linguistically motivated?
  • Wu (1997) conjectured no
  • we will answer it empirically
How to find good binarizations?
Naïve Algorithm
Naïve Algorithm

- Try all binary bracketings
  - finds all possible binarizations
  - exponential complexity (Catalan number)
Naïve Algorithm

- Try all binary bracketings
  - finds all possible binarizations
  - exponential complexity (Catalan number)
Naïve Algorithm

- Try all binary bracketings
  - finds all possible binarizations
  - exponential complexity (Catalan number)
- However
  - we need a much faster algorithm!
  - one synchronous binarization is enough
  - we prefer unique solution for each permutation
    - sharing sub-binarizations
Linear-time Algorithm

• like shift-reduce, just need one left-to-right scan

• each iteration
  • shifts one number from input to stack
  • keep reducing the pair at the top of the stack

1 5 3 4 2

1 5

1 5

?
Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

```
1  5  3  4  2
  5  3  ?
1  5  3
```

```
5
4
3
2
1
?
```

```
5
4
3
2
1
?
```

```
1  5  3  4  2
1  5  3  4  2
1  5  3  4  2
1  5  3  4  2
1  5  3  4  2
```
Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

1 5 3 4 2

1 5 3 4

?
Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

```
1  5  3  4  2
1  5  3-4
1  5  3-4  2
```
Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

```
1  5  3  4  2
1  5  3-4  ?
```

Diagram:
```
+---+---+---+---+---+
| 5 |   |   |   |   |
| 4 | 3 |   |   |   |
| 2 |   |   |   |   |
+---+---+---+---+---+
  1  5  3-4  2
  1   1   ?
  1   5  3-4  2
```
Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

```
1  5  3  4  2
```

```
1  3-5
```

```
<table>
<thead>
<tr>
<th>5</th>
<th>3</th>
<th>3-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3-5</td>
<td>2</td>
</tr>
</tbody>
</table>
```
Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

```
1 5 3 4 2
1 3-5
```

```
<table>
<thead>
<tr>
<th>5</th>
<th></th>
<th></th>
<th>3-5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>3-5</td>
<td>2</td>
</tr>
</tbody>
</table>
```
Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

```
5 1 3 4 2
1 3-5 2
1 3-5 2
1
```
Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

1  5  3  4  2
1  2-5
1  1  2-5
1  2-5
Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

```
5 1 2

1 1
1
? 2-5
```
Linear-time Algorithm

- like shift-reduce, just need one left-to-right scan
- each iteration
  - shifts one number from input to stack
  - keep reducing the pair at the top of the stack

1 5 3 4 2

1-5

1-5

1-5

1-5
Properties of the Algorithm

- runs in $O(n)$ time
  - $n$ shifts and at most $(n-1)$ reductions
- the permutation is synchronously binarizable
  - iff. it is reduced to exactly one element at the end
- and in such case, the algorithm will return
  - the unique left-heavy binarization tree
  - corresponding to the non-ambiguous ITG (Wu, 1997)
Experiments
ISI syntax-based MT system (Galley et al., 2004)
- based on tree-to-string rules
- English-side subtrees and Chinese-side strings
- when decoding, internally flattened to SCFGs

\[ S \rightarrow x_1 \ x_3 \ x_2 \]

\[ S \rightarrow \text{NP VP PP}, \ \text{NP PP VP} \]
Data

• training data
  • English/Chinese parallel corpus
    • 2.9 M sentence pairs (50 M English words)
    • word-aligned by GIZA++ (symmetrized by “union”)
  • English side parsed by a variant of Collins parser
  • 50.8 M tree-to-string rules extracted

• test data
  • 116 short sentences (<= 16 words) from NIST 2002
Binarizability
Among 50.8 M rules, only 0.3 % are non-binarizable
Among 50.8 M rules, only 0.3 % are non-binarizable

although theoretically this ratio is much higher
Among 50.8 M rules, only 0.3 % are non-binarizable

- although theoretically this ratio is much higher
- identical decoding results without these non-bin rules
Better Decoding

- decoding is based on the CKY algorithm
- BLEU scores at various beam settings
- synchronous binarization consistently better

![Graph showing BLEU scores and number of edges proposed during decoding]

<table>
<thead>
<tr>
<th>Binarization</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>monolingual</td>
<td>36.25</td>
</tr>
<tr>
<td>synchronous</td>
<td>38.44</td>
</tr>
</tbody>
</table>

(∼ CPU time)
Examples
Examples

- the last step is for them to receive new job training.
Examples

- the last step is for them to receive new job training.
- the final step is to receive new work training.
Examples

- the last step is for them to receive new job training.
- the final step is to receive new work training.
- the final step is to receive new job training.
Examples

- the last step is for them to receive new job training.
- the final step is to receive new work training.
- the final step is to receive new job training.
Examples

- the last step is for them to receive new job training.
- the final step is to receive new work training.
- the final step is to receive new job training.
- turkey has agreed in principle to take over from britain the command of the peace-keeping troops in afghanistan.
Examples

• the last step is for them to receive new job training.
• the final step is to receive new work training.
• the final step is to receive new job training.

• turkey has agreed in principle to take over from britain the command of the peace-keeping troops in afghanistan.
• turkey already agreed in principle that the rebels from the british took over the command of afghan peacekeeping force.
Examples

- the last step is for them to receive new job training.
- the final step is to receive new work training.
- the final step is to receive new job training.

- turkey has agreed in principle to take over from britain the command of the peace-keeping troops in afghanistan.
- turkey already agreed in principle that the rebels from the british took over the command of afghan peacekeeping force.
- turkey has agreed in principle to take command of afghan peacekeeping force there from britain.
Conclusion and Future Work
Conclusion and Future Work

- Theoretical side
  - a theory on binarization for synchronous grammars
    - theorems and proofs in the paper
  - an efficient algorithm for binarizing permutations
Conclusion and Future Work

- Theoretical side
  - a theory on binarization for synchronous grammars
    - theorems and proofs in the paper
  - an efficient algorithm for binarizing permutations

- Practical side
  - syntactic re-orderings are largely binarizable
  - binarized grammars improve translation quality
Conclusion and Future Work

- Theoretical side
  - a theory on binarization for synchronous grammars
    - theorems and proofs in the paper
  - an efficient algorithm for binarizing permutations
- Practical side
  - syntactic re-orderings are largely binarizable
  - binarized grammars improve translation quality
- Future work: extend to more powerful formalisms
  - e.g., synchronous tree-adjoining grammars (STAGs)
Thank you!