Binarized Forest-to-Tree Translation

Authors
Hao Zhang  Google Research
Licheng Fang  University of Rochester
Peng Xu  Google Research
Xiaoyun Wu  Google Research

Presenter
Waleed Ammar
synchronous tree substitution grammar

Phrase based rule: {“green houses”,
     “maisons vertes”}

STSG rule1: {np(adj(green) n(house)),
         np(n(maisons) adj(vertes))}

STSG rule2: {np(adj₀ n₁), np(n₁ adj₀ )}

STSG rule3: {np(adj₀ n₁), “{1} {0}”}
optimal forest-to-string translation

$$e = \mathcal{Y}( \underset{d \in D(T), \ T \in F(f)}{\arg \max} \ P(d|T) )$$

\[
\begin{align*}
 f & \quad := \text{source sentence} \\
 F(f) & \quad := \text{forest of parse trees of a source sentence} \\
 D(T) & \quad := \text{synchronous derivations of a parse tree} \\
 e & \quad := \text{best translation} \\
 Y & \quad := \text{yield of a derivation}
\end{align*}
\]
parse-forest construction

rule extraction

rule binarization
parse-forest construction

• Traditional forests
  – Construction
  – Drawbacks

• Proposed forests
  – Construction
  – Merits
best parse

```plaintext
(0) det
(1) n
(2) np
(3) v
(4) n
(5) s

the
frogs
ate
soup
```
parse-forest construction

CYK-1
binarization:
ancestor
annotation

(0)det₂ (2)np₅ (5)s
(1)n₂
(3)v₅
(4)n₅

the
frogs
ate
soup
parse-forest construction

<table>
<thead>
<tr>
<th>(0)det₂</th>
<th>(2)np₅</th>
<th>(5)s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>frogs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>soup</td>
<td></td>
</tr>
</tbody>
</table>

### CYK-1

**binarization:**

- drop edges
parse-forest construction

CYK-1 binarization:

<table>
<thead>
<tr>
<th></th>
<th>(0)det$_2$</th>
<th>(2)np$_5$</th>
<th>(5)s</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)n$_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3)v$_5$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4)n$_5$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

processing order

- the
- frogs
- ate
- soup
parse-forest construction

CYK-1
binarization:

processing
at each node:

left & right have
shared ancestor?
Yes!

the

frogs

ate

soup
parse-forest construction

**CYK-1**

binarization:

processing at each node:

left & right have shared ancestor? No!
parse-forest construction

CYK-1 binarization:

processing at each node:

left & right have shared ancestor? Yes!

The diagram shows a parse forest with the following nodes:

- (0)det₂
- (1)n₂
- (2)np₅
- (3)v₅
- (4)n₅
- (5)s

The words are:
- the
- frogs
- ate
- soup
### parse-forest construction

**CYK-1 binarization:**

<table>
<thead>
<tr>
<th></th>
<th>(0)det₂</th>
<th>(2)np₅</th>
<th>(5)s</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>the</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>frogs</strong></td>
<td>(1)n₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ate</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>soup</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Processing at each node:**

- Left & right have shared ancestor?
  - Yes!
parse-forest construction

CYK-1
binarization:

processing
at each node:

left & right have
shared ancestor?
No!

the
frogs
ate
soup

(0)det₂
(1)n₂
(2)np₅
(3)v₅
(4)n₅
(5)s
(6)v_n₅
### CYK-1 binarization:

**processing at each node:**

- left & right have shared ancestor? Yes!

#### Example:

<table>
<thead>
<tr>
<th></th>
<th>(0)det₂</th>
<th>(2)np₅</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>frogs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>soup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1)n₂</td>
<td>(3)v₅</td>
<td>(6)v_n₅</td>
<td>(5)s</td>
</tr>
</tbody>
</table>

The diagram shows the parse-forest construction with the words "the frogs ate soup."
parse-forest construction

CYK-1 binarization:

processing at each node:

left & right have shared ancestor? Yes!
parse-forest construction

CYK-1 binarization:
processing at each node:
left & right have shared ancestor?
No!
parse-forest construction

CYK-1 binarization:

processing at each node:
left & right have shared ancestor?
No!
parse-forest construction

CYK-1
binarization:
processing
at each node:

left & right have
shared ancestor?
Yes!

the

(0)det$_2$

(2)np$_5$

(7)np_v$_5$

(5)s

frogs

(1)n$_2$

(3)v$_5$

(6)v_n$_5$

ate

(4)n$_5$

soup
parse-forest construction

CYK-1 binarization:

processing at each node:

left & right have shared ancestor? Yes!
parse-forest construction

CYK-1
binarization:

final
parsing
forest
The best parse for the sentence "the frogs ate soup" is a forest construction that captures the structure of the sentence.
parse-forest construction

CYK-2 binarization:
ancestor annotation

(0)det$_{2,5}$ (1)n$_{2,5}$ (2)np$_5$ (3)v$_5$ (4)n$_5$ (5)s

the frogs ate soup
### CYK-2

**Binarization:**

Processing at each node:

Left & right have shared ancestor? Yes!

<table>
<thead>
<tr>
<th></th>
<th>(0)det_{2,5}</th>
<th>(2)np_5</th>
<th>(5)s</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>n_{2,5}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(8)n_v_5</td>
<td>(3)v_5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>frogs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ate</td>
<td></td>
<td>(4)n_5</td>
</tr>
<tr>
<td>5</td>
<td>soup</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
parse-forest construction

CYK-2
binarization:
final
parsing
forest

(0)det_{2,5} → (6)v_{n_5} 
(1)n_{2,5} → (4)n_5 
(2)np_5 → (3)v_5 
(7)np_{v_5} → (5)s 
(8)n_{v_5} → (6)v_{n_5} 
(9)n_{v_5_{n_5}}
the frogs ate soup

best parse

CYK-1 forest

CYK-2 forest
parse-forest construction

(a)

(b)

CYK-2
parse-forest construction
parse-forest construction
parse-forest construction

rule extraction

rule binarization
i.e. Which tree fragments to use?

English: The frogs ate soup

Hindi: मेंढ़कों नें सूप खा लिया
synchronous tree substitution grammar

Phrase based rule: {“green houses”,
                      “maisons vertes”}

STSG rule1: \{\text{np(adj(green) n(house))},
               \text{np(n(maisons) adj(vertes))}\}

STSG rule2: \{\text{np(adj}_0 \text{n}_1), \text{np(n}_1 \text{adj}_0 )\}

STSG rule3: \{\text{np(adj}_0 \text{n}_1), \text{"\{1\} \{0\}"}\}
rule extraction

1. Mark admissible nodes

English: The frogs ate soup

Hindi: मेंढ़कों नें सूप खा लिया
<table>
<thead>
<tr>
<th>Seed fragment</th>
<th>Src span</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0-0</td>
</tr>
</tbody>
</table>

2. Build tree fragments

English: The frogs ate soup

Hindi: मेंढ़कों ने सूप खा लिया
rule extraction

<table>
<thead>
<tr>
<th>Seed fragment</th>
<th>Src span</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>0-0</td>
</tr>
<tr>
<td>det</td>
<td>0-0</td>
</tr>
<tr>
<td>frogs</td>
<td>1-1</td>
</tr>
<tr>
<td>n</td>
<td>1-1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>v_n</td>
<td>2-3</td>
</tr>
<tr>
<td>s</td>
<td>0-3</td>
</tr>
</tbody>
</table>

English: The frogs ate soup

Hindi: मेंढ़कों नें सूप खा लिया
rule extraction

<table>
<thead>
<tr>
<th>Composed fragment</th>
<th>Src span</th>
</tr>
</thead>
<tbody>
<tr>
<td>det(the)</td>
<td>0-0</td>
</tr>
<tr>
<td>n(frogs)</td>
<td>1-1</td>
</tr>
<tr>
<td>np(det n)</td>
<td>0-1</td>
</tr>
<tr>
<td>np(det(the) n)</td>
<td>0-1</td>
</tr>
<tr>
<td>np(det n(frogs))</td>
<td>0-1</td>
</tr>
<tr>
<td>np(det(the) n(frogs))</td>
<td>0-1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

English: The frogs ate soup

Hindi: मेंढ़कों ने सूप खा लिया
The frogs ate soup

Hindi: मेंढ़कों ने सूप खा लिया
rule extraction

<table>
<thead>
<tr>
<th>Composed fragment</th>
<th>Src span</th>
</tr>
</thead>
<tbody>
<tr>
<td>det(the)</td>
<td>0-0</td>
</tr>
<tr>
<td>n(frogs)</td>
<td>1-1</td>
</tr>
<tr>
<td>np(det n)</td>
<td>0-1</td>
</tr>
<tr>
<td>np(det the n)</td>
<td>0-1</td>
</tr>
<tr>
<td>np(det the n(frogs))</td>
<td>0-1</td>
</tr>
</tbody>
</table>

Hmm... Maybe we need to prune

English: The frogs ate soup
Hindi: मेंढ़कों नें सूप खा लिया
2. Build tree fragments

- Each node maintains a K-best list
- Score(rule) = (height, frontiers, terminals)
- Score(r1+r2) = (max{h1+h2}+1, f1+f2, t1+t2)
- Importance: Height → Frontiers → Terminals
Figure 2: Tree-to-string rule composition as cube-pruning. The left shows two lists of composed rules sorted by their geometric measures (height, # frontiers, # frontier terminals), under the gluing rule of VP → VBD VP – C. The right part shows a cube view of the combination space. We explore the space from the top-left corner to the neighbors.
3. Enumerate and match rules for the tree fragments at admissible nodes.
parse-forest construction

rule extraction

rule binarization
• Rules are not necessarily binary
• More sharing
• Cube pruning
• Runtime vs. offline binarization

Figure 4: Synchronous binarization for a tree-to-string rule. The top rule can be binarized into two smaller rules.
experiments

• Compare to phrase-based and hiero
• 16 rules per node
• Shift-reduce dep. parser (87.8% labeled, 88.8% u)
• Map dependency trees to constituent trees
## experiments

<table>
<thead>
<tr>
<th>Source Words</th>
<th>Target Words</th>
<th>BLEU dev</th>
<th>BLEU test</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-Chinese</td>
<td>287M</td>
<td>254M</td>
<td>29.7</td>
</tr>
<tr>
<td>English-Czech</td>
<td>66M</td>
<td>57M</td>
<td>31.7</td>
</tr>
<tr>
<td>English-French</td>
<td>857M</td>
<td>996M</td>
<td>31.9</td>
</tr>
<tr>
<td>English-German</td>
<td>45M</td>
<td>43M</td>
<td>-</td>
</tr>
<tr>
<td>English-Spanish</td>
<td>216M</td>
<td>238M</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: The Sizes of Parallel Texts.

<table>
<thead>
<tr>
<th>Source Words</th>
<th>BLEU rules</th>
<th>BLEU dev</th>
<th>BLEU test</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-Chinese</td>
<td>no binarization</td>
<td>378M</td>
<td>28.0</td>
</tr>
<tr>
<td>English-Czech</td>
<td>head-out</td>
<td>408M</td>
<td>30.0</td>
</tr>
<tr>
<td>English-French</td>
<td>cyk-1</td>
<td>527M</td>
<td>31.6</td>
</tr>
<tr>
<td>English-German</td>
<td>cyk-2</td>
<td>803M</td>
<td>31.9</td>
</tr>
<tr>
<td>English-Spanish</td>
<td>cyk-3</td>
<td>1053M</td>
<td>32.0</td>
</tr>
<tr>
<td>English-Spanish</td>
<td>cyk-∞</td>
<td>1441M</td>
<td>32.0</td>
</tr>
</tbody>
</table>

Table 3: Comparing different source tree binarization schemes for English-Chinese translation, showing both BLEU scores and model sizes. The rule counts include normal phrases which are used at the leaf level during decoding.

Table 2: Translation results comparing bf2s, the binarized-forest-to-string system, pb, the phrase-based system, and hier, the hierarchical phrase-based system.
experiments

- Optimal degree for CYK-n

<table>
<thead>
<tr>
<th>binarization</th>
<th>rules</th>
<th>BLEU dev</th>
<th>BLEU test</th>
</tr>
</thead>
<tbody>
<tr>
<td>no binarization</td>
<td>378M</td>
<td>28.0</td>
<td>36.3</td>
</tr>
<tr>
<td>head-out</td>
<td>408M</td>
<td>30.0</td>
<td>38.2</td>
</tr>
<tr>
<td>cyk-1</td>
<td>527M</td>
<td>31.6</td>
<td>40.5</td>
</tr>
<tr>
<td>cyk-2</td>
<td>803M</td>
<td>31.9</td>
<td>40.7</td>
</tr>
<tr>
<td>cyk-3</td>
<td>1053M</td>
<td>32.0</td>
<td>40.6</td>
</tr>
<tr>
<td>cyk-∞</td>
<td>1441M</td>
<td>32.0</td>
<td>40.3</td>
</tr>
</tbody>
</table>

Table 3: Comparing different source tree binarization schemes for English-Chinese translation, showing both BLEU scores and model sizes. The rule counts include normal phrases which are used at the leaf level during decoding.
experiments

• Binarized forests vs. parse-generated forests

<table>
<thead>
<tr>
<th></th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev</td>
</tr>
<tr>
<td>cyk-2</td>
<td>14.9</td>
</tr>
<tr>
<td>parser</td>
<td>14.7</td>
</tr>
</tbody>
</table>

Table 4: Binarized forests versus parser-generated forests for forest-to-string English-German translation.
experiments

- Effect of rule binarization

<table>
<thead>
<tr>
<th></th>
<th>BLEU</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev</td>
<td>test</td>
</tr>
<tr>
<td>head-out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cube pruning</td>
<td>29.2</td>
<td>37.0</td>
</tr>
<tr>
<td>+ synch. binarization</td>
<td>30.0</td>
<td>38.2</td>
</tr>
<tr>
<td>cyk-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cube pruning</td>
<td>31.7</td>
<td>40.5</td>
</tr>
<tr>
<td>+ synch. binarization</td>
<td>31.9</td>
<td>40.7</td>
</tr>
</tbody>
</table>

Table 5: The effect of synchronous binarization for tree-to-string and forest-to-string systems, on the English-Chinese task.
discussion