Hierarchical Phrase-Based Translation with Weighted Finite State Transducers

Gonzalo Iglesias$^1$ Adrià de Gispert$^2$
Eduardo R. Banga$^1$ William Byrne$^2$

$^1$Department of Signal Processing and Communications
University of Vigo, Spain

$^2$Department of Engineering.
University of Cambridge, U.K.

NAACL-HLT 2009, Boulder, Colorado
Outline

1. Hierarchical Translation with WFSTs
   - Introducing HiFST
   - Example
   - Lattice Construction over the CYK grid
   - Delayed Translation
   - Pruning

2. Translation Experiments
Outline

1. Hierarchical Translation with WFSTs
   - Introducing HiFST
   - Example
   - Lattice Construction over the CYK grid
   - Delayed Translation
   - Pruning

2. Translation Experiments
Introducing HiFST I

HiFST: New hierarchical decoder that uses lattices (WFSTs) rather than k-best lists

Why use Lattices instead of k-best lists?

- Compactness and Efficiency
- Semiring Operations
  - rmepsilon, determinize, minimize, compose, prune shortestpath, ...
- WFSTs: OpenFST, available at openfst.org (Allauzen et al. 2007)
Introducing HiFST II

- classical CYK algorithm: source side, hypotheses recombination, no pruning
  - Given a sentence $s_1 \ldots s_J$, find out every derivation starting at cell $(S, 1, J)$
- Lattices $\mathcal{L}(N, x, y)$ are built for each cell following back-pointers of the grid
  - Objective is lattice $\mathcal{L}(S, 1, J)$, at the top of the grid
Outline

1. Hierarchical Translation with WFSTs
   - Introducing HiFST
   - Example
   - Lattice Construction over the CYK grid
   - Delayed Translation
   - Pruning

2. Translation Experiments
Example I

- Consider sentence $s_1 s_2 s_3$
- We are looking for $L(S, 1, 3)$
  - Represents all the translations generated by derivations covering span $s_1 s_2 s_3$
- Toy grammar:

  $R^1$: $X \rightarrow \langle s_1 s_2 s_3, t_1 t_2 \rangle$
  $R^2$: $X \rightarrow \langle s_1 s_2, t_7 t_8 \rangle$
  $R^3$: $X \rightarrow \langle s_3, t_9 \rangle$
  $R^4$: $S \rightarrow \langle X, X \rangle$
  $R^5$: $S \rightarrow \langle S X, S X \rangle$
  $R^6$: $X \rightarrow \langle s_1, t_{20} \rangle$
  $R^7$: $X \rightarrow \langle X_1 s_2 X_2 X_1, t_{10} X_2 \rangle$
  $R^8$: $X \rightarrow \langle X_1 s_2 X_2 X_2, t_{10} X_1 \rangle$
Example II
Phrase-based Scenario

Introducing HiFST
Example
Lattice Construction over the CYK grid
Delayed Translation
Pruning

Hierarchical Translation with WFSTs
Translation Experiments
Summary

\[
R^1: \ X \rightarrow \langle s_1 s_2 s_3, t_1 t_2 \rangle \\
R^2: \ X \rightarrow \langle s_1 s_2, t_7 t_8 \rangle \\
R^3: \ X \rightarrow \langle s_3, t_9 \rangle \\
R^4: \ S \rightarrow \langle X, X \rangle \\
R^5: \ S \rightarrow \langle S, X, S, X \rangle \\
\]

\[
R^1:\ y \quad R^2:\ y \quad R^3:\ y \quad R^4:\ y \\
\]

\[
R^5:\ y \quad R^2:\ y \quad R^3:\ y \\
\]

\[
R^1:\ y \quad R^2:\ y \\
\]

\[
R^5:\ y \quad R^2:\ y \\
\]
Hierarchical Translation with WFSTs

Translation Experiments

Summary

Introducing HiFST

Example

Lattice Construction over the CYK grid

Delayed Translation

Pruning

Example III

Hierarchical Scenario

\[ R^3: X \rightarrow \langle s_3, t_9 \rangle \]
\[ R^4: S \rightarrow \langle X, X \rangle \]
\[ R^6: X \rightarrow \langle s_1, t_{20} \rangle \]
\[ R^7: X \rightarrow \langle X_1 s_2 X_2, X_1 t_{10} X_2 \rangle \]
\[ R^8: X \rightarrow \langle X_1 s_2 X_2, X_2 t_{10} X_1 \rangle \]
Rule lattices are merged (i.e. with union) into one single (top) cell lattice
Lattice Construction over the CYK grid

A **cell lattice** is a union of rule lattices:

\[
\mathcal{L}(N, x, y) = \bigoplus_{r \in R(N, x, y)} \mathcal{L}(N, x, y, r)
\]

(1)

A **rule lattice** is a concatenation of element lattices:

\[
\mathcal{L}(N, x, y, r) = \bigotimes_{i=1}^{\left|\alpha^r\right|} \mathcal{L}(N, x, y, r, i)
\]

(2)

An **element lattice** may be a simple arc binding two states (terminal, i.e., word) or a sublattice (non-terminal):

\[
\mathcal{L}(N, x, y, r, i) = \begin{cases} 
\mathcal{A}(\alpha_i) & \text{if } \alpha_i \in T \\
\mathcal{L}(N', x', y') & \text{else}
\end{cases}
\]

(3)
A procedure for lattice translation

```
function buildFst(N, x, y)
    if \( \mathcal{L}(N, x, y) \) exists, return \( \mathcal{L}(N, x, y) \)
    for each rule applied in cell \( (N, x, y) \),
        for each element in rule
            if element is a word, create \( \mathcal{A}(element) \)
            else buildFst(backpointers(element))
    Create rule lattice by catenation of element lattices
    Create cell lattice \( \mathcal{L}(N, x, y) \) by unioning rule lattices
    Reduce \( \mathcal{L}(N, x, y) \) with semiring operations and return
```
1. Hierarchical Translation with WFSTs
   - Introducing HiFST
   - Example
   - Lattice Construction over the CYK grid
   - Delayed Translation
   - Pruning

2. Translation Experiments
Delayed Translation I

- As the algorithm goes up the grid, lattices grow in complexity
  - Severe memory and speed problems
- Solution: Delay translation using unique pointers to sublattices → skeleton lattices
- Once the building procedure has finished, i.e. $\mathcal{L}(S, 1, J)$ has been built, just expand it...
  - Substituting recursively each special unique pointer by appropriate sublattice
Delayed Translation II

lattices with translated text and pointers to lower lattices produced by hierarchical rules

pointers to lattices at lower cells

lattices with translated text

CYK grid
Easily implemented. Formally, we define $g(N, x, y)$ as the unique pointer for a given cell. Then change Equation 3:

$$L(N, x, y, r, i) = \begin{cases} \mathcal{A}(\alpha_i) & \text{if } \alpha_i \in T \\ L(N', x', y') & \text{else} \end{cases}$$

into:

$$L(N, x, y, r, i) = \begin{cases} \mathcal{A}(\alpha_i) & \text{if } \alpha_i \in T \\ \mathcal{A}(g(N', x', y')) & \text{else} \end{cases}$$ (4)
Hierarchical Translation with WFSTs
Translation Experiments
Summary

Introducing HiFST
Example
Lattice Construction over the CYK grid
Delayed Translation
Pruning

Delayed Translation IV

Usual operations (rmepsilon, determinize, minimize, etc) still work!
Reduction of lattice size
Outline

1. Hierarchical Translation with WFSTs
   - Introducing HiFST
   - Example
   - Lattice Construction over the CYK grid
   - Delayed Translation
   - Pruning

2. Translation Experiments
Pruning

- Final translation lattice $L(S, 1, J)$ typically requires pruning
  - Compose with Language Model of target words
  - Perform likelihood-based pruning (Allauzen et al 2007)

- Pruning in Search:
  - If number of states, non-terminal category and source span meet certain conditions, then:
    - Expand Pointers in translation Lattice and Compose with Language Model
    - Perform likelihood-based pruning of the lattice
    - Remove Language Model
1. Hierarchical Translation with WFSTs
   - Introducing HiFST
   - Example
   - Lattice Construction over the CYK grid
   - Delayed Translation
   - Pruning

2. Translation Experiments
Translation Experiments I

- HCP: Hierarchical Cube Pruning decoder, k-best=10000
- NIST MT08 Arabic-to-English and Chinese-to-English translation tasks.
  - Hiero Shallow for Arabic (Iglesias et al. 2009)
  - Hiero Full for Chinese
- MET optimization done in the usual way with n-best lists.
  Features:
  - target language model
  - translation models, lexical models
  - word and rule penalties, glue rule
  - three rule count features (Bender et al. 2007)
Translation Experiments II

- English LM: 4-gram over 965 million word subset English Gigaword Third Edition

- Rescoring steps:
  - Large-LM rescoring of 10000-best list with 5-gram zero cut-off stupid back-off language models (T. Brants et al. 2007)
    - ~4.7B words of English nw, vocabulary used based on the phrases covered by the parallel text
    - Implemented with WFSTs (failure transitions)
  - Minimum Bayes Risk (MBR). Rescore 1000-best hyps
Translation Experiments III
AR → EN

- No pruning in search → speed increased, HCP search errors: 18%
- Richer search space: increased gains from 5Gram LM + Minimum Bayes Risk rescoring

<table>
<thead>
<tr>
<th>decoder</th>
<th>mt02-05-tune</th>
<th></th>
<th>mt02-05-test</th>
<th></th>
<th>mt08</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLEU</td>
<td>TER</td>
<td>BLEU</td>
<td>TER</td>
<td>BLEU</td>
<td>TER</td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCP</td>
<td>52.2</td>
<td>41.6</td>
<td>51.5</td>
<td>42.2</td>
<td>42.5</td>
<td>48.6</td>
</tr>
<tr>
<td>+5g+MBR</td>
<td>53.2</td>
<td>40.8</td>
<td>52.6</td>
<td>41.4</td>
<td>43.4</td>
<td>48.1</td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HiFST</td>
<td>52.2</td>
<td>41.5</td>
<td>51.6</td>
<td>42.1</td>
<td>42.4</td>
<td>48.7</td>
</tr>
<tr>
<td>+5g+MBR</td>
<td>53.7</td>
<td>40.4</td>
<td>53.3</td>
<td>40.9</td>
<td>44.0</td>
<td>48.0</td>
</tr>
</tbody>
</table>
Hierarchical Translation with WFSTs
Translation Experiments
Summary

Translation Experiments IV
ZH→EN

- More efficient search: 48% reduction in search errors
- HCP improves if using HiFST MET parameters (b)
- HiFST is comparable to HCP in first pass
- HiFST produces richer/better hypotheses for rescoring

<table>
<thead>
<tr>
<th>decoder</th>
<th>MET</th>
<th>tune-nw</th>
<th>test-nw</th>
<th>mt08</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BLEU</td>
<td>TER</td>
<td>BLEU</td>
</tr>
<tr>
<td>a</td>
<td>HCP</td>
<td>HCP</td>
<td>31.6</td>
<td>59.7</td>
</tr>
<tr>
<td>b</td>
<td>HCP</td>
<td>HiFST</td>
<td>31.7</td>
<td>60.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32.4</td>
<td>59.2</td>
</tr>
<tr>
<td>c</td>
<td>HiFST</td>
<td>HiFST</td>
<td>32.0</td>
<td>60.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32.9</td>
<td>58.4</td>
</tr>
</tbody>
</table>
Summary

- We have introduced HiFST, a new hierarchical decoder based on WFSTs
  - Easy to implement, as complexity is hidden by OpenFST library
- Delayed translation effectively reduces complexity during lattice construction
- Pruning in search is completely avoided for AR → EN, yielding a very fast translation
- ZH → EN requires pruning, but it is more selective than HCP (i.e. fired by non-terminal, number of states, span, etc)
Fewer search errors in the k-best translation hypotheses improves rescoring and MBR

MET parameter optimization is improved using HiFST

HiFST will be available to download soon
Thank you!

Questions?