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

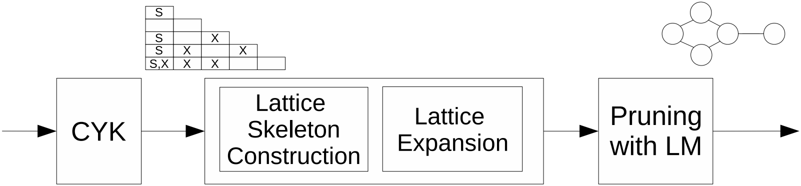
2. Translation Experiments

Outline

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

Example I

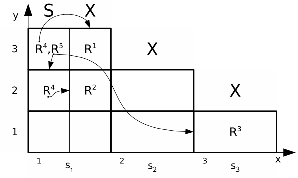
- Consider sentence $s_1 s_2 s_3$
- We are looking for $\mathcal{L}(S, 1, 3)$
  - Represents all the translations generated by derivations covering span $s_1 s_2 s_3$
- Toy grammar:

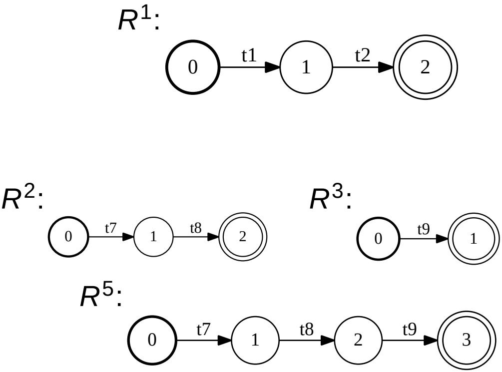
$$
\begin{aligned}
R^1&: X \rightarrow \langle \mathrm{s}_1\ \mathrm{s}_2\ \mathrm{s}_3, \mathrm{t}_1\ \mathrm{t}_2 \rangle \\
R^2&: X \rightarrow \langle \mathrm{s}_1\ \mathrm{s}_2, \mathrm{t}_7\ \mathrm{t}_8 \rangle \\
R^3&: X \rightarrow \langle \mathrm{s}_3, \mathrm{t}_9 \rangle \\
R^4&: S \rightarrow \langle X, X \rangle \\
R^5&: S \rightarrow \langle S\ X, S\ X \rangle \\
R^6&: X \rightarrow \langle \mathrm{s}_1, \mathrm{t}_{20} \rangle \\
R^7&: X \rightarrow \langle \mathrm{X}_1\ \mathrm{s}_2\ \mathrm{X}_2, \mathrm{X}_1\ \mathrm{t}_{10}\ \mathrm{X}_2 \rangle \\
R^8&: X \rightarrow \langle \mathrm{X}_1\ \mathrm{s}_2\ \mathrm{X}_2, \mathrm{X}_2\ \mathrm{t}_{10}\ \mathrm{X}_1 \rangle
\end{aligned}
$$

Example II

Phrase-based Scenario

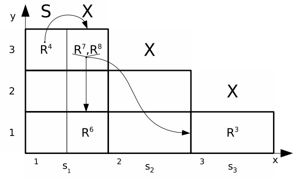
R^1: $X \rightarrow \langle s_1\ s_2\ s_3, t_1\ t_2 \rangle$
R^2: $X \rightarrow \langle s_1\ s_2, t_7\ t_8 \rangle$
R^3: $X \rightarrow \langle s_3, t_9 \rangle$
R^4: $S \rightarrow \langle X, X \rangle$
R^5: $S \rightarrow \langle S\ X, S\ X \rangle$

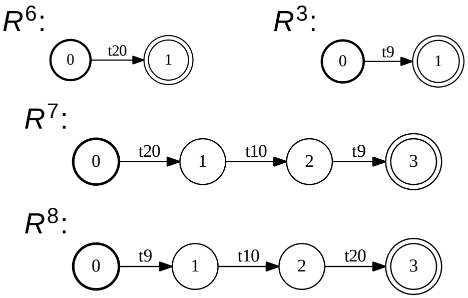




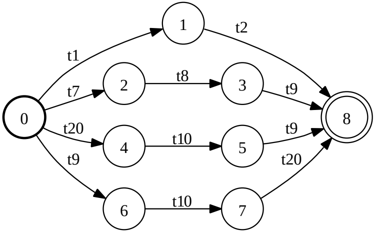
Example III
Hierarchical Scenario

R^3: $X \longrightarrow \langle s_3, t_9 \rangle$
R^4: $S \longrightarrow \langle X, X \rangle$
R^6: $X \longrightarrow \langle s_1, t_{20} \rangle$
R^7: $X \longrightarrow \langle X_1\ s_2\ X_2, X_1\ t_{10}\ X_2 \rangle$
R^8: $X \longrightarrow \langle X_1\ s_2\ X_2, X_2\ t_{10}\ X_1 \rangle$





Example IV
Cell Lattice



- Rule lattices are merged (i.e. with union) into one single (top) cell lattice

Outline

Lattice Construction over the CYK grid

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

$$\mathcal{L}(N,x,y) = \bigoplus_{r \in R(N,x,y)} \mathcal{L}(N,x,y,r) \quad (1)$$

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

$$\mathcal{L}(N,x,y,r) = \bigotimes_{i=1..|\alpha^r|} \mathcal{L}(N,x,y,r,i) \quad (2)$$

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

$$\mathcal{L}(N,x,y,r,i) = \begin{cases} \mathcal{A}(\alpha_i) & \text{if } \alpha_i \in \mathbf{T} \\ \mathcal{L}(N',x',y') & \text{else} \end{cases} \quad (3)$$

A procedure for lattice translation

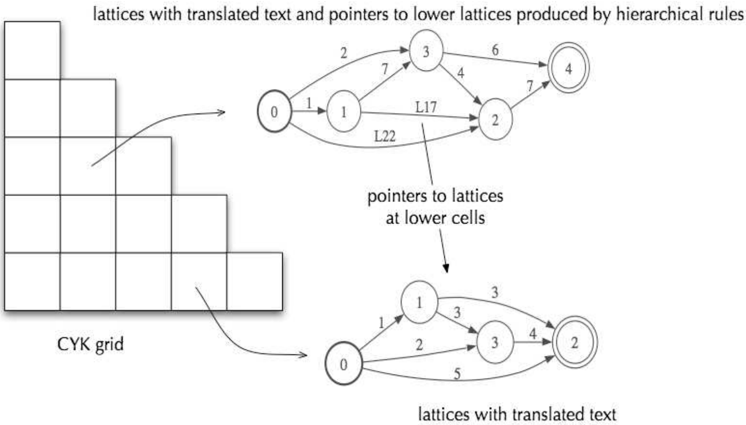
1 **function** buildFst(N,x,y)
2   if $\mathcal{L}(N,x,y)$ exists, return $\mathcal{L}(N,x,y)$
3   for each rule applied in cell (N,x,y),
4   for each element in rule
5   if element is a word, create $\mathcal{A}(element)$
6-8   else buildFst(backpointers(element))
9   Create rule lattice by catenation of element lattices
10   Create cell lattice $\mathcal{L}(N,x,y)$ by unioning rule lattices
11-14   Reduce $\mathcal{L}(N,x,y)$ with semiring operations and return

Outline

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 $\rightarrow$ 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



Delayed Translation III

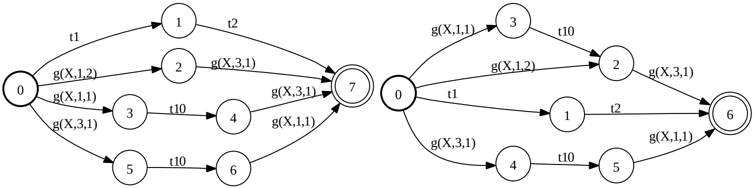
Easily implemented. Formally, we define $g(N,x,y)$ as the unique pointer for a given cell. Then change Equation 3:

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

into:

$$\mathcal{L}(N,x,y,r,i) = \begin{cases} \mathcal{A}(\alpha_i) & \text{if } \alpha_i \in \mathbf{T} \\ \mathcal{A}(g(N',x',y')) & \text{else} \end{cases} \tag{4}$$

Delayed Translation IV



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

Outline

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

Outline

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)
    - $\sim$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

	decoder	*mt02-05-tune*		*mt02-05-test*		*mt08*	
		BLEU	TER	BLEU	TER	BLEU	TER
a	HCP	52.2	41.6	51.5	42.2	42.5	48.6
	+5g+MBR	53.2	40.8	52.6	41.4	43.4	48.1
b	HiFST	52.2	41.5	51.6	42.1	42.4	48.7
	+5g+MBR	53.7	40.4	53.3	40.9	44.0	48.0

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

	decoder	MET	*tune-nw*		*test-nw*		*mt08*	
			BLEU	TER	BLEU	TER	BLEU	TER
a	HCP	HCP	31.6	59.7	31.9	59.7	–	–
b	HCP	HiFST	31.7	60.0	32.2	59.9	27.2	60.2
	+5g+MBR		32.4	59.2	32.7	59.4	28.1	59.3
c	HiFST	HiFST	32.0	60.1	32.2	60.0	27.1	60.5
	+5g+MBR		32.9	58.4	33.4	58.5	28.9	58.9

Summary I

- 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 $AR \rightarrow EN$, yielding a very fast translation
- $ZH \rightarrow EN$ requires pruning, but it is more selective than HCP (i.e. fired by non-terminal, number of states, span, etc)

Summary II

- 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?