The HiFST System for the EuroParl Spanish-to-English Task

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.

SEPLN 2009, Donostia

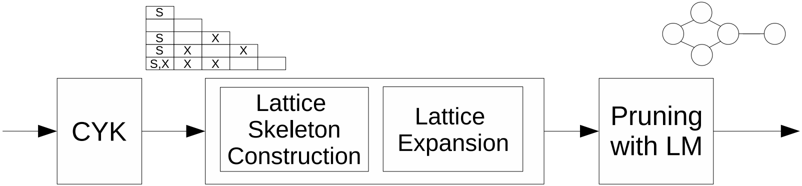
Outline

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



- variant of CYK algorithm on SCFG: source side, hypotheses recombination, no pruning
  - Given a sentence $s_1 ... s_J$, 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

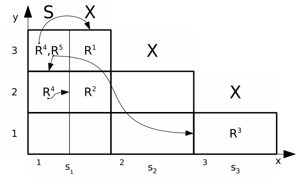
- Consider sentence $s_1 s_2 s_3$
- We are looking for $\mathcal{L}(S,1,3)$
  - Represents all the translations generated by derivations covering span $s_1 s_2 s_3$
- Toy grammar:

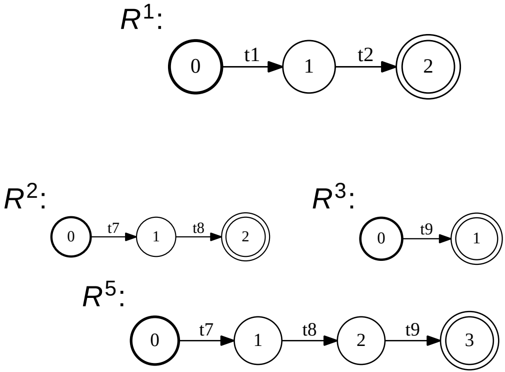
$$
\begin{aligned}
R^1&: X \rightarrow \langle \mathrm{s}_1\ \mathrm{s}_2\ \mathrm{s}_3, \mathrm{t}_1\ \mathrm{t}_2 \rangle \\
R^2&: X \rightarrow \langle \mathrm{s}_1\ \mathrm{s}_2, \mathrm{t}_7\ \mathrm{t}_8 \rangle \\
R^3&: X \rightarrow \langle \mathrm{s}_3, \mathrm{t}_9 \rangle \\
R^4&: S \rightarrow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

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: $S \rightarrow \langle S\ X, S\ X \rangle$

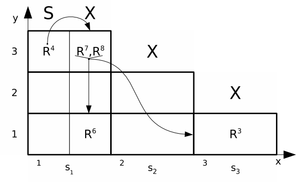


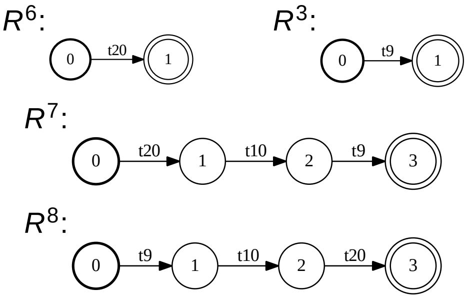


Example III

Hierarchical Scenario

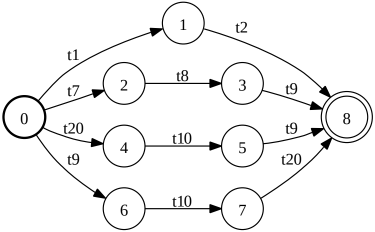
R^3: $X \rightarrow \langle s_3, t_9 \rangle$
R^4: $S \rightarrow \langle X, X \rangle$
R^6: $X \rightarrow \langle s_1, t_{20} \rangle$
R^7: $X \rightarrow \langle X_1\ s_2\ X_2, X_1\ t_{10}\ X_2 \rangle$
R^8: $X \rightarrow \langle X_1\ s_2\ X_2, X_2\ t_{10}\ X_1 \rangle$





Example IV

Cell Lattice



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

Outline

Hierarchical Translation with WFSTs

- Introducing HiFST
- Example
- Delayed Translation
- Pruning
- Null Words

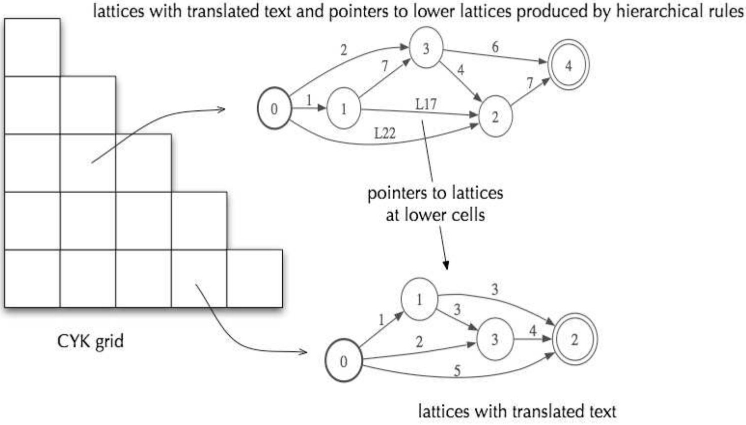
Translation Experiments

- Experimental Setup
- Full versus Shallow
- Filtering by Number of Translations
- Contrastive Experiments with Patterns
- Rescoring

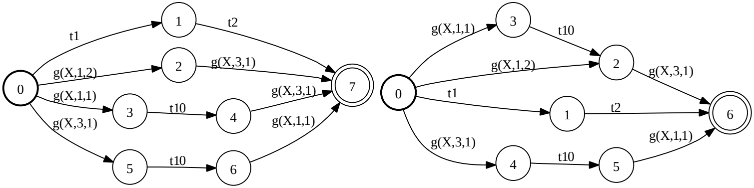
Delayed Translation I

- 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



- 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

Null Words

- SMT systems tend to benefit from allowing small number of deletions
- One deletion rule for each source-language word
- Huge increase in the search-space
- Limited by composition with following FST:

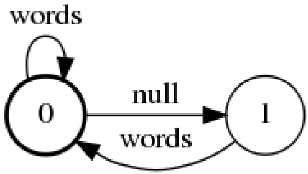


Figure: Consecutive Null filtering.

Outline

Experimental Setup I

- Spanish-to-English translation shared task of the ACL 2008 Workshop on Statistical Machine Translation, WMT
- 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)
- English LM: 4-gram over 965 million word subset English Gigaword Third Edition

Experimental Setup II

Excluded Rules	Types
$\langle X_1w,X_1w\rangle$, $\langle wX_1,wX_1\rangle$	1530797
$\langle X_1wX_2,*\rangle$	737024
$\langle X_1wX_2w,X_1wX_2w\rangle$, $\langle wX_1wX_2,wX_1wX_2\rangle$	41600246
$\langle wX_1wX_2w,*\rangle$	45162093
$N_{nt}.N_e$= 1.3 mincount=5	39013887
$N_{nt}.N_e$= 2.4 mincount=10	6836855

Table: Rules excluded from initial grammar extraction.

- Rule pattern: replacing every sequence of terminals by ‘w’
- Patterns classified by number of non-terminals N_{nt} and elements N_e(non-terminals and substrings of terminals). **5** classes: $N_{nt}.N_e = 1.2, 1.3, 2.3, 2.4, 2.5$.

Outline

Full versus Shallow

- Full grammar: $X \rightarrow \langle\gamma,\alpha\rangle$, $\gamma, \alpha \in (\{X\} \cup \mathbf{T})^+$
- Shallow grammar: $X \rightarrow \langle\gamma_s,\alpha_s\rangle$, $X \rightarrow \langle V,V\rangle$, $V \rightarrow \langle s,t\rangle$

Hiero Model	*dev2006*	*test2008*
Shallow	33.65/7.852	33.65/7.877
Full	33.63/7.849	33.66/7.880

Table: Performance of Hiero Full versus Hiero Shallow Grammars.

- Shallow: more constrained search-space, but exact – much faster decoding times

Outline

Filtering by Number of Translations

NT	*dev2006*	*test2008*
20	33.65/7.852	33.65/7.877
30	33.61/7.849	33.75/7.896
40	33.63/7.853	33.73/7.883

Table: Performance of baseline grammar when varying the filter by number of translations per source-side.

Outline

Contrastive Experiments with Patterns

- $G1$: shallow grammar with $NT = 20$ (baseline).
- $G2$: (subset) mincount=5 for $N_{nt}.N_e = 2.3$ and $N_{nt}.N_e = 2.5$
- $G3$: (superset) add $\langle X_1 w, X_1 w \rangle$

	dev2006	*test2008*
$G1$	33.65/7.852	33.65/7.877
$G2$	33.47/7.838	33.65/7.877
$G3$	33.09/7.787	33.14/7.808

Table: Contrastive performance with $G1$,$G2$,$G3$.

Outline

Rescoring I

- Shallow grammar with $NT = 30$
- *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

Rescoring II

	dev2006	*test2008*
HiFST	33.61/7.849	33.75/7.896
+5gram	33.66 /7.902	33.90/7.954
+MBR	33.87 /7.901	34.24/7.962

Table: EuroParl Spanish-to-English translation results (lower-cased IBM BLEU / NIST) after MET and subsequent rescoring steps

Summary

- HiFST is a new hierarchical decoder based on WFSTs with state-of-the-art perfomance
    - 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 $SP \rightarrow EN$, yielding a very fast translation
- HiFST will be available to download soon

Thank you!

Questions?