

# Hierarchical Phrase-based Translation Representations

**Gonzalo Iglesias**<sup>\*</sup>, Cyril Allauzen<sup>‡</sup>, William Byrne<sup>\*</sup>  
Adrià de Gispert<sup>\*</sup>, Michael Riley<sup>‡</sup>

<sup>\*</sup>Cambridge University Engineering Department

<sup>‡</sup>Google Research

Edinburgh, 29th July 2011



Google research

# Hierarchical Phrase-Based Decoding I

Given:

- ▶ A source sentence  $s$
- ▶ A Synchronous Context Free Grammar (SCFG)  $G$
- ▶ An n-gram Language Model  $M$

Exact decoding is done in three general steps:

1. *Apply the grammar to the sentence:  $\mathcal{T} = \{s\} \circ G$*
2. *Intersect the Search Space  $\mathcal{T}$  with the language model:  $\mathcal{L} = \mathcal{T} \cap M$ ,*
3. *Search for the highest-probability path:  $\operatorname{argmax} \mathcal{L}$*

## Hierarchical Phrase-Based Decoding II

- ▶ Different **hierarchical phrase-based translation representations** of the space of translation hypotheses  $\mathcal{T}$ :
  - ▶ Hypergraphs: Cube Pruning Decoder<sup>1</sup>
  - ▶ Finite-State Automata(FSA): **HiFST**, a weighted finite-state hiero decoder<sup>2</sup>.
    - ▶ Exact Search for carefully designed hiero grammars
- ▶ This paper: We propose to use Push-Down Automata (PDA) in the decoder (**HiPDT**) for exact decoding of larger grammars.

---

<sup>1</sup>David Chiang. **Hierarchical phrase-based translation**. Computational Linguistics, 2007. 33(2), pp201-228.

<sup>2</sup>G. Iglesias et al. **Hierarchical phrase-based translation with weighted finite state transducers**. In Proceedings of NAACL 2009, pp433-441.

# Push-Down Automata I

- ▶ Informally: PDA augment FSA with a stack
- ▶ PDT extension<sup>3</sup> implemented in OpenFST<sup>4</sup>.
  - ▶ We restrict a transition to be labeled by a stack operation or a regular input symbol but not both.
  - ▶ Stack operations are **implicitly** represented by pairs of open and close "parentheses"
  - ▶ This representation is identical to the finite automaton representation except that certain symbols (the parentheses) have special semantics.
  - ▶ Advantage: many FSA operations still work or do so with minor changes

---

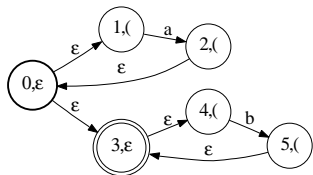
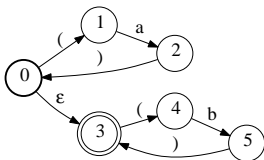
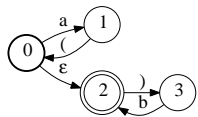
<sup>3</sup>Cyril Allauzen and Michael Riley. **Pushdown Transducers**. <http://pdt.openfst.org>, 2011.

<sup>4</sup>Cyril Allauzen et al. **OpenFst: A General and Efficient Weighted Finite-State Transducer Library**. In Proceedings of CIAA, 2007.

## Push-Down Automata II

- ▶ A *weighted pushdown automaton* (PDA)  $T$  over the tropical semiring  $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$  is a 9-tuple  $(\Sigma, \Pi, \bar{\Pi}, Q, E, I, F, \rho)$  where  $\Sigma$  is the finite input alphabet,  $\Pi$  **and**  $\bar{\Pi}$  **are the finite open and close parenthesis alphabets**,  $Q$  is a finite set of states,  $I \in Q$  the initial state,  $F \subseteq Q$  the set of final states,  $E \subseteq Q \times (\Sigma \cup \hat{\Pi} \cup \{\epsilon\}) \times (\mathbb{R} \cup \{\infty\}) \times Q$  a finite set of transitions, and  $\rho : F \rightarrow \mathbb{R} \cup \{\infty\}$  the final weight function.
  
- ▶ A *weighted finite state automaton* (FSA) is a particular case of a PDA where the open and close parentheses alphabets are empty.

## Push-Down Automata III



PDA accepting  $a^n b^n$

PDA accepting  $a^* b^*$ .

Equivalent FSA

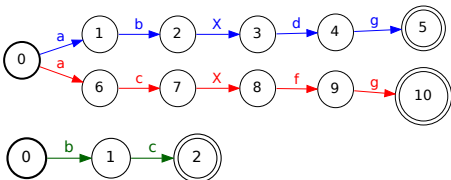
- ▶ An accepting path  $\pi$  is **balanced** if its sequence of open/close parentheses is in a Dyck Language  $( [ ( ) ( ) ] \{ \} [ ] ) ( )$
- ▶ Accepts paths with finite number of unmatched open parentheses – bounded stack
- ▶ There exists an equivalent FSA.

## Replacement

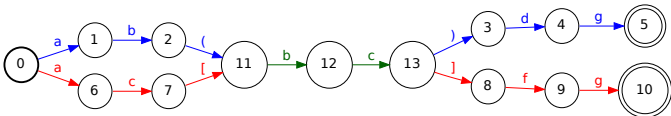
The replacement algorithm transforms a Recursive Transition Network (RTN) into an equivalent PDA

► RTN:

$S \rightarrow a b X d g$   
 $S \rightarrow a c X f g$   
 $X \rightarrow b c$



► PDA:



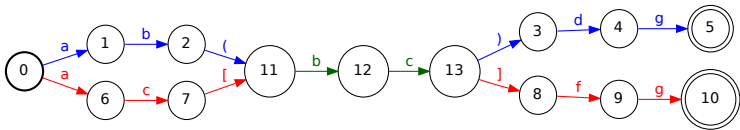
- The replacement algorithm transforms an RTN into an equivalent PDA.
- The RTN has finite recursion level  $\rightarrow$  PDA guaranteed to have a bounded stack.

# Intersection

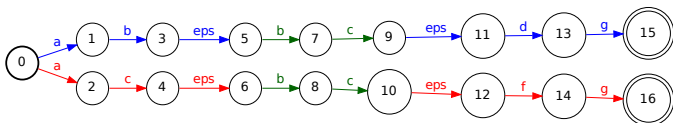
- ▶ PDA  $T$  intersection with FSA  $M$  is closed (Bar-Hillel intersection)
- ▶ Almost identical to FSA intersection
  - ▶ parentheses treated as epsilons but retained as parentheses in the result
- ▶ Time/Space complexity:  $O(|T||M|)$

## Expansion

- ▶ If the PDA is stack-bounded, it represents a regular language  $\rightarrow$  The PDA can be expanded into an equivalent FSA.
- ▶ PDA:



- ▶ FSA:



# Shortest Distance

SHORTESTDISTANCE( $T$ )

```

1  for each  $q \in Q$  and  $a \in \Pi$  do
2     $B[q, a] \leftarrow \emptyset$ 
3  GETDISTANCE( $T, I$ )
4  return  $d[f, I]$ 

```

RELAX( $q, s, w, \mathcal{S}$ )

```

1  if  $d[q, s] > w$  then
2     $d[q, s] \leftarrow w$ 
3  if  $q \notin \mathcal{S}$  then
4    ENQUEUE( $\mathcal{S}, q$ )

```

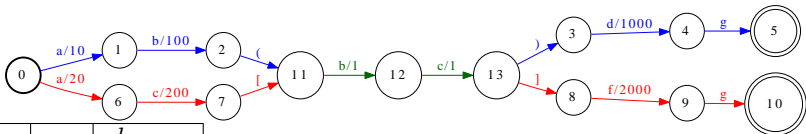
GETDISTANCE( $T, s$ )

```

1  for each  $q \in Q$  do
2     $d[q, s] \leftarrow \infty$ 
3   $d[s, s] \leftarrow 0$ 
4   $\mathcal{S}_s \leftarrow s$ 
5  while  $\mathcal{S}_s \neq \emptyset$  do
6     $q \leftarrow \text{HEAD}(\mathcal{S}_s)$ 
7    DEQUEUE( $\mathcal{S}_s$ )
8  for each  $e \in E[q]$  do
9    if  $i[e] \in \Sigma \cup \{\epsilon\}$  then
10     RELAX( $n[e], s, d[q, s] + w[e], \mathcal{S}_s$ )
11   elseif  $\overline{i[e]} \in \overline{\Pi}$  then
12      $B[s, \overline{i[e]}] \leftarrow B[s, \overline{i[e]}] \cup \{e\}$ 
13   elseif  $i[e] \in \Pi$  then
14     if  $d[n[e], n[e]]$  is undefined then
15       GETDISTANCE( $T, n[e]$ )
16     for each  $e' \in B[n[e], i[e]]$  do
17        $w \leftarrow d[q, s] + w[e] + d[p[e'], n[e]] -$ 
18       RELAX( $n[e'], s, w, \mathcal{S}_s$ )

```

# Shortest Distance



$s_1$	$s_2$	$d_{s_1 \rightarrow s_2}$
0	0	0
0	1	10
0	2	110
11	11	0
11	12	1
11	13	2
0	3	112
...		
0	6	20
0	7	220
0	8	222
...		

- ▶ Memoization of shortest distance for local sub-lattices
- ▶ For a general PDA, complexity  $O(|T|^3)$
- ▶ If PDA derived from acyclic RTN  $\rightarrow$  complexity  $O(|T|)$
- ▶ But if PDA intersected with  $M$  this introduces  $|M|^3$  in time complexity

# Hiero with RTN/FSA

► **HiFST decoder**<sup>5</sup>: →

Time/Space:  $O(e^{|s|^3|G|} |M|)$

1. CYK Parse sentence  $s$  with Grammar  $G$ .
2. Build RTN
3. Expand RTN to FSA
4. Intersect FSA with Language Model  $M$
5. Shortest Path / prune for lattice rescoring

**(finite)**

**(no cyclic references)**

---

<sup>5</sup>Adrià de Gispert et al. **Hierarchical Phrase-based Translation with Weighted Finite State Transducers and Shallow-N Grammars**. Computational Linguistics, 2010.

## Hiero with PDA

- ▶ **HiPDT decoder:**  $\longrightarrow$  Time:  $O(|s|^3|G||M|^3)$ ; Space:  $O(|s|^3|G||M|^2)$ 
  1. CYK Parse sentence  $s$  with Grammar  $G$ . **(finite)**
  2. Build RTN **(no cyclic references)**
  3. Convert RTN into PDA (Replacement algorithm) **(stack-bounded)**
  4. Intersect PDA with  $\mathbf{M}$
  5. PDA-Shortest Path / (Pruned) Expansion of PDA for lattice rescoring
- ▶ **CFG/Hypergraph decoder:**  $\longrightarrow$  Time,Space:  $O(|s|^3|G||M|^3)$
- ▶ **HiFST decoder:**  $\longrightarrow$  Time,Space:  $O(e^{|s|^3|G|}|M|)$
- ▶ In practice, HiFST and HiPDT faster due to **optimizations** over RTN
- ▶ HiPDT will be more efficient than HiFST for bigger grammars, if language model is small enough.
- ▶ HiFST more efficient with bigger language models and smaller grammar

# Decoding Pipeline with Entropy-Pruned LMs I

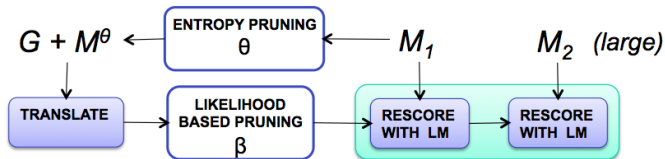
- ▶ Given:
  1. Hierarchical Grammar  $G$
  2. Language Model  $M_1$  for decoding
  3. Large 5-gram Language Model  $M_2$  for rescoring
- ▶ We entropy-prune<sup>6</sup>  $M_1$  under relative perplexity  $\theta$  to obtain  $M^\theta$  for direct translation + rescoring with  $M_1$
- ▶ Successful in Speech Recognition systems<sup>7</sup>

---

<sup>6</sup>Andreas Stolcke. **Entropy-based Pruning of Backoff Language Models**. Proceedings of DARPA Broadcast News Transcription and Understanding Workshop, 1998.

<sup>7</sup>Andrej Ljolje et al. **Efficient general lattice generation and rescoring**. In Proceedings of Eurospeech, 1999.

## Decoding Pipeline with Entropy-Pruned LMs II



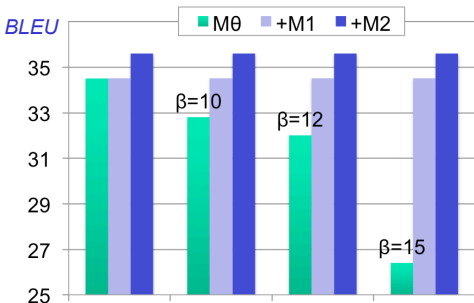
1. Fast translation under  $M^\theta$
2. Likelihood based pruning of output lattice, beam width  $\beta$
3. Take out  $M^\theta$  scores from lattice
4. Rescore lattice with  $M_1$
5. Additionally, rescore with large 5-gram model  $M_2$

# Zh→En Translation with Compact Grammars

## ▶ Compact Grammar $G_1$

- ▶ Only rules with translation probability  $> 0.01$  are used
- ▶ Entire lattice can be expanded and intersected with  $M_1$
- ▶ FSA (HiFST) and PDA (HiPDT) representations equally good
- ▶ Exact decoding – we can analyse impact of different entropy pruned language models

- ▶ Full performance recovered after rescoring with LM
- ▶ Critical beam width  $\beta$  required
- ▶ Decoding speed-up



Entropy threshold  $\theta$     -    1.0E-09    1.0E-08    1.0E-07

Number of N-grams:    200M    20M    4M    1M

Time (sec/w):    0.68    0.38    0.28    0.20

# Zh→En Translation with Large Grammars

- ▶ Large Grammar  $G_2$ 
  - ▶ All observed rules are considered (+alternatives per rule)
  - ▶ HiFST representation cannot decode under 10GB
    - ▶ RTN expansion is critical (pruning in search would alleviate)
  - ▶ HiPDT achieves exact decoding under small  $M_\theta$

entropy pruning $\theta$	HiFST			HiPDT		
	Success	Expand Fails	Intersect Fails	Success	Intersect Fails	Expand Fails
$10^{-9}$	12%	51%	37%	40%	8%	52%
$10^{-8}$	16%	53%	31%	76%	1%	23%
$10^{-7}$	18%	53%	29%	<b>99.8%</b>	0%	0.2%

- ▶ Improved results with HiPDT (+0.5 BLEU) due to exact decoding with larger grammar  $G_2$ .

## Conclusions

- ▶ HiPDT allows exact decoding of larger hierarchical grammars than HiFST, but with smaller language models – Improves translation performance
- ▶ Expensive PDA shortest path algorithm after PDA intersection with LM
- ▶ Entropy-pruned LMs allow *faster decoding times*, less memory requirements. Same performance after LM rescoring.
- ▶ Translation search space is finite – RTN/PDA/FSA efficient representations
- ▶ HiPDT (and HiFST) implemented with general purpose library OpenFST<sup>8</sup> – complexity is hidden to the user
- ▶ Hybrid FSA/PDA approach for improved robustness (exact decoding under  $M$  when feasible)
- ▶ Other LM smoothing strategies<sup>9</sup>

---

<sup>8</sup>See [www.openfst.org](http://www.openfst.org)

<sup>9</sup>Chelba et al. **Study on Interaction between Entropy Pruning and Kneser-Ney Smoothing**. In Proceedings of Interspeech 2010.

Thank you!

Questions?