Hierarchical Phrase-based Translation Representations

Bill Byrne

Department of Engineering
University of Cambridge

16 July 2013

Work done jointly with:

Gonzalo Iglesias and Adrià de Gispert, University of Cambridge

Cyril Allauzen and Michael Riley, Google Research

Overview[1] [2]

1. Comparison of ‘Representations’ of translation hypotheses produced with stochastic synchronous context-free grammars
   - CFGs / Hypergraphs
   - Finite State automata (FSAs) / Recursive Transition Networks (RTNs)
   - Push-down Automata (PDAs)
2. Some analysis of impact of representation on search procedures
3. Search procedures for PDAs specialised for SMT
4. Some results in speed/quality/pruning in translation

---

[1] Hierarchical phrase-based translation representations. G. Iglesias, C. Allauzen, W. Byrne, A.de Gispert, M. Riley. EMNLP 2011

[2] C. Allauzen, W. Byrne, A. de Gispert, G. Iglesias, M. Riley. *Pushdown Automata in Statistical Machine Translation*. submitted to Computational Linguistics

Why Study FSMs in Machine Translation

Large, complex translation grammars can lead to search errors in translation

- Search error: whenever the decoder returns something other than the top-scoring hypothesis under the translation grammar and language model

Search errors complicate the modelling problem

- Translations produced are not necessarily those intended in grammar construction
- Difficult to talk about grammars independently of a decoder architecture

Goal: Decoders which can handle complicated grammars and are less prone to search errors

- Better infrastructure for exploring translation grammars

$\Rightarrow$ FSMs are still very useful, even for translation with SCFGs

Hierarchical Phrase Based Translation [3]

- Context free bi-grammar
- A single non-terminal symbol
- Productions include a mix of non-terminals and terminals
  - Word translations
    - X→⟨maison , house⟩
  - Phrasal translations
    - X→⟨daba una bofetada , slap⟩
  - Mixed
    - X→⟨X bleue , blue X⟩
    - X→⟨X1 X2 , X2 of X1⟩
  - 'Glue' rules
    - S→⟨S X , S X⟩
    - S→⟨X , X⟩

[3] Chiang, David. 2007. Hierarchical phrase-based translation. Computational Linguistics, 33(2):201228.

Hierarchical Phrase-based Translation

R_1: S→⟨X , X⟩
R_2: S→⟨S X , S X⟩
R_3: X→⟨s_1 , said⟩
R_4: X→⟨s_1 s_2 , the president said⟩
R_5: X→⟨s_1 s_2 s_3 , Syrian president says⟩
R_6: X→⟨s_2 , president⟩
R_7: X→⟨s_3 , the Syrian⟩
R_8: X→⟨s_4 , yesterday⟩
R_9: X→⟨s_5 , that⟩
R_{10}: X→⟨s_6 , would return⟩
R_{11}: X→⟨s_6 , he would return⟩

s_1	s_2	s_3	s_4	s_5	s_6
wqAl	**Alr}ys**	**Alswry**	**Ams**	**Anh**	**syEwd**

(وقال الرئيس السوري امس انه سيعود)

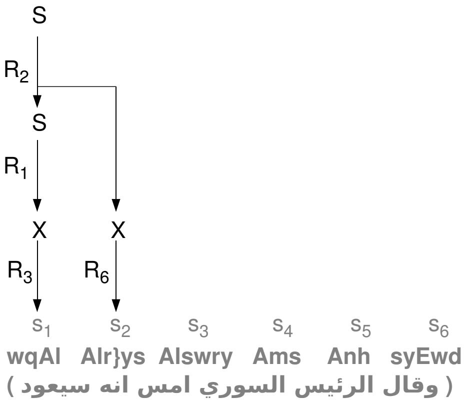
Hierarchical Phrase-based Translation

said

R_1: $\mathbf{S \rightarrow \langle X\ ,\ X \rangle}$
R_2: $S \rightarrow \langle S\ X\ ,\ S\ X \rangle$
R_3: $\mathbf{X \rightarrow \langle s_1\ ,\ \text{said} \rangle}$
R_4: $X \rightarrow \langle s_1\ s_2\ ,\ \text{the president said} \rangle$
R_5: $X \rightarrow \langle s_1\ s_2\ s_3\ ,\ \text{Syrian president says} \rangle$
R_6: $X \rightarrow \langle s_2\ ,\ \text{president} \rangle$
R_7: $X \rightarrow \langle s_3\ ,\ \text{the Syrian} \rangle$
R_8: $X \rightarrow \langle s_4\ ,\ \text{yesterday} \rangle$
R_9: $X \rightarrow \langle s_5\ ,\ \text{that} \rangle$
R_{10}: $X \rightarrow \langle s_6\ ,\ \text{would return} \rangle$
R_{11}: $X \rightarrow \langle s_6\ ,\ \text{he would return} \rangle$

Hierarchical Phrase-based Translation

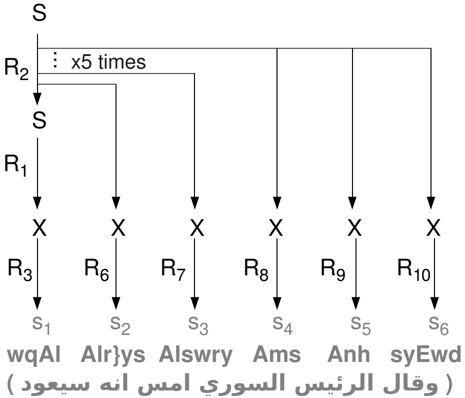
said president



R_1: $S \rightarrow \langle X\ ,\ X \rangle$
R_2: $\mathbf{S \rightarrow \langle S\ X\ ,\ S\ X \rangle}$
R_3: $X \rightarrow \langle s_1$, said$\rangle$
R_4: $X \rightarrow \langle s_1\ s_2$, the president said$\rangle$
R_5: $X \rightarrow \langle s_1\ s_2\ s_3$, Syrian president says$\rangle$
R_6: $\mathbf{X \rightarrow \langle s_2}$ **, president**$\rangle$
R_7: $X \rightarrow \langle s_3$, the Syrian$\rangle$
R_8: $X \rightarrow \langle s_4$, yesterday$\rangle$
R_9: $X \rightarrow \langle s_5$, that$\rangle$
R_{10}: $X \rightarrow \langle s_6$, would return$\rangle$
R_{11}: $X \rightarrow \langle s_6$, he would return$\rangle$

Hierarchical Phrase-based Translation

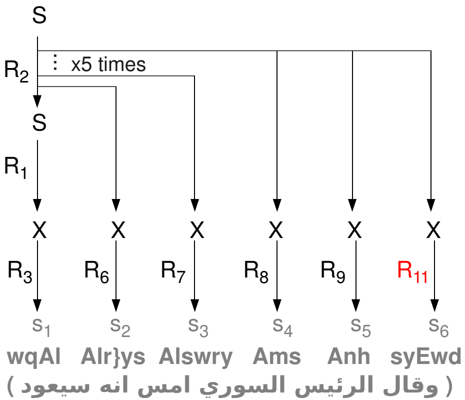
said president the Syrian yesterday that would return



R_1: $S \rightarrow \langle X \;,\; X \rangle$
R_2: $\mathbf{S \rightarrow \langle S\, X \;,\; S\, X \rangle}$
R_3: $X \rightarrow \langle s_1 \;,\; \text{said} \rangle$
R_4: $X \rightarrow \langle s_1\, s_2 \;,\; \text{the president said} \rangle$
R_5: $X \rightarrow \langle s_1\, s_2\, s_3 \;,\; \text{Syrian president says} \rangle$
R_6: $X \rightarrow \langle s_2 \;,\; \text{president} \rangle$
R_7: $\mathbf{X \rightarrow \langle s_3 \;,\; \textbf{the Syrian} \rangle}$
R_8: $\mathbf{X \rightarrow \langle s_4 \;,\; \textbf{yesterday} \rangle}$
R_9: $\mathbf{X \rightarrow \langle s_5 \;,\; \textbf{that} \rangle}$
R_{10}: $\mathbf{X \rightarrow \langle s_6 \;,\; \textbf{would return} \rangle}$
R_{11}: $X \rightarrow \langle s_6 \;,\; \text{he would return} \rangle$

Hierarchical Phrase-based Translation

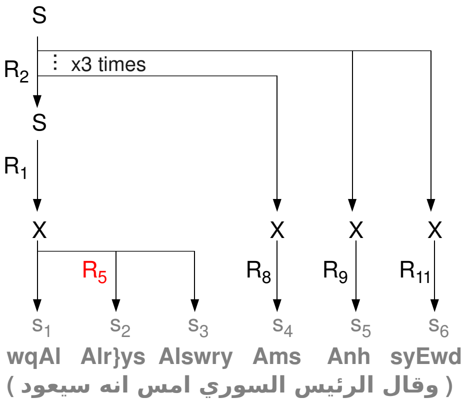
said president the Syrian yesterday that he would return



R_1: S→⟨X , X⟩
R_2: S→⟨S X , S X⟩
R_3: X→⟨s_1 , said⟩
R_4: X→⟨s_1 s_2 , the president said⟩
R_5: X→⟨s_1 s_2 s_3 , Syrian president says⟩
R_6: X→⟨s_2 , president⟩
R_7: X→⟨s_3 , the Syrian⟩
R_8: X→⟨s_4 , yesterday⟩
R_9: X→⟨s_5 , that⟩
R_{10}: X→⟨s_6 , would return⟩
R_{11}: X→⟨s_6 , he would return⟩

Hierarchical Phrase-based Translation

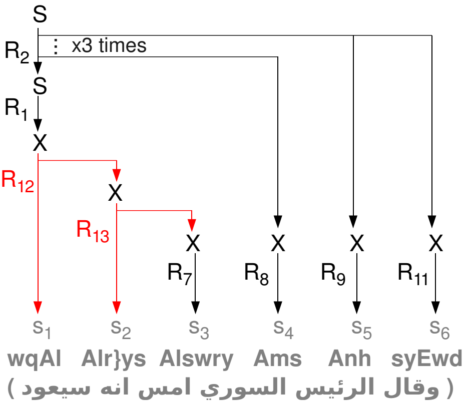
Syrian president says yesterday that he would return



R_1: $S \rightarrow \langle X\ ,\ X \rangle$
R_2: $S \rightarrow \langle S\ X\ ,\ S\ X \rangle$
R_3: $X \rightarrow \langle s_1$, said$\rangle$
R_4: $X \rightarrow \langle s_1\ s_2$, the president said$\rangle$
R_5: **$X \rightarrow \langle s_1\ s_2\ s_3$, Syrian president says$\rangle$**
R_6: $X \rightarrow \langle s_2$, president$\rangle$
R_7: $X \rightarrow \langle s_3$, the Syrian$\rangle$
R_8: $X \rightarrow \langle s_4$, yesterday$\rangle$
R_9: $X \rightarrow \langle s_5$, that$\rangle$
R_{10}: $X \rightarrow \langle s_6$, would return$\rangle$
R_{11}: $X \rightarrow \langle s_6$, he would return$\rangle$

Hierarchical Phrase-based Translation (2)

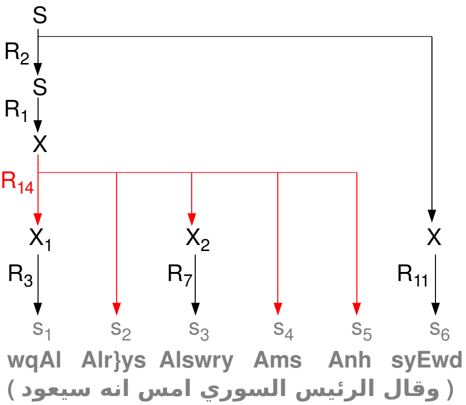
the Syrian president said yesterday that he would return



R_1: $S \rightarrow \langle X \ , \ X \rangle$
R_2: $S \rightarrow \langle S\ X \ , \ S\ X \rangle$
R_3: $X \rightarrow \langle s_1 \ , \ \text{said} \rangle$
...
R_6: $X \rightarrow \langle s_2 \ , \ \text{president} \rangle$
R_7: $X \rightarrow \langle s_3 \ , \ \text{the Syrian} \rangle$
R_8: $X \rightarrow \langle s_4 \ , \ \text{yesterday} \rangle$
R_9: $X \rightarrow \langle s_5 \ , \ \text{that} \rangle$
R_{10}: $X \rightarrow \langle s_6 \ , \ \text{would return} \rangle$
R_{11}: $X \rightarrow \langle s_6 \ , \ \text{he would return} \rangle$
$\mathbf{R_{12}}$: $X \rightarrow \langle s_1\ X \ , \ X\ \text{said} \rangle$
$\mathbf{R_{13}}$: $X \rightarrow \langle s_2\ X \ , \ X\ \text{president} \rangle$

Hierarchical Phrase-based Translation (2)

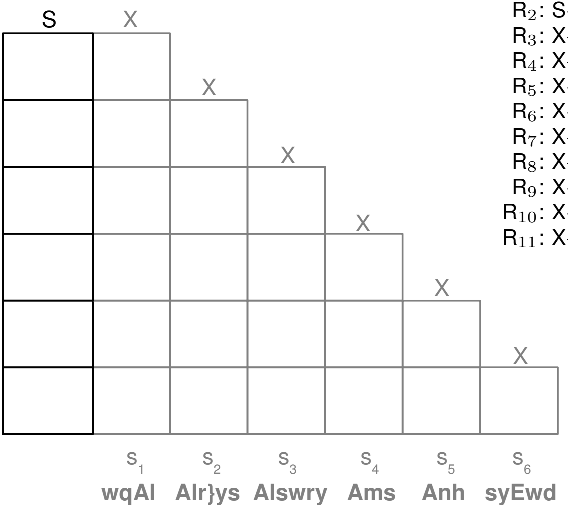
yesterday the Syrian president said that he would return



R_1: $S \rightarrow \langle X\ ,\ X \rangle$
R_2: $S \rightarrow \langle S\ X\ ,\ S\ X \rangle$
R_3: $X \rightarrow \langle s_1\ ,\ \text{said} \rangle$
...
R_6: $X \rightarrow \langle s_2\ ,\ \text{president} \rangle$
R_7: $X \rightarrow \langle s_3\ ,\ \text{the Syrian} \rangle$
R_8: $X \rightarrow \langle s_4\ ,\ \text{yesterday} \rangle$
R_9: $X \rightarrow \langle s_5\ ,\ \text{that} \rangle$
R_{10}: $X \rightarrow \langle s_6\ ,\ \text{would return} \rangle$
R_{11}: $X \rightarrow \langle s_6\ ,\ \text{he would return} \rangle$
R_{14}: $X \rightarrow \langle X_1\ s_2\ X_2\ s_4\ s_5\ ,$ y'day X_2 president X_1 that$\rangle$

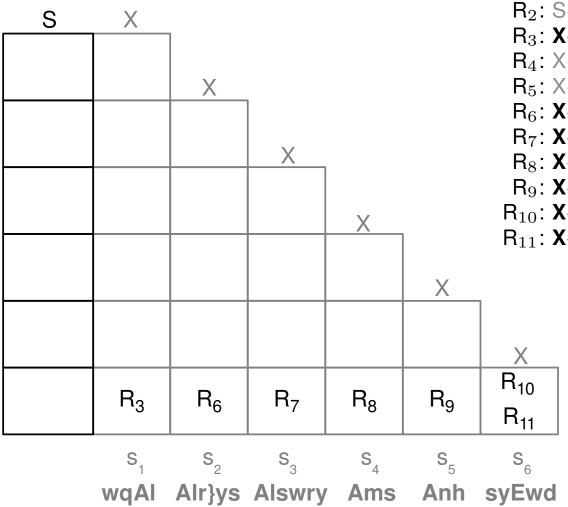
- Each rule has a probability assigned by the Translation Model

Keeping Track of All Derivations. CYK Grid



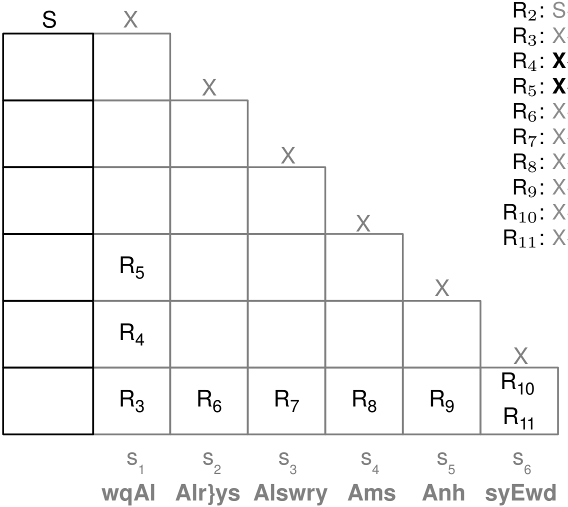
R_1: $S \rightarrow \langle X\ ,\ X \rangle$
R_2: $S \rightarrow \langle S\ X\ ,\ S\ X \rangle$
R_3: $X \rightarrow \langle s_1$, said$\rangle$
R_4: $X \rightarrow \langle s_1\ s_2$, the president said$\rangle$
R_5: $X \rightarrow \langle s_1\ s_2\ s_3$, Syrian president says$\rangle$
R_6: $X \rightarrow \langle s_2$, president$\rangle$
R_7: $X \rightarrow \langle s_3$, the Syrian$\rangle$
R_8: $X \rightarrow \langle s_4$, yesterday$\rangle$
R_9: $X \rightarrow \langle s_5$, that$\rangle$
R_{10}: $X \rightarrow \langle s_6$, would return$\rangle$
R_{11}: $X \rightarrow \langle s_6$, he would return$\rangle$

Keeping Track of All Derivations. CYK Grid



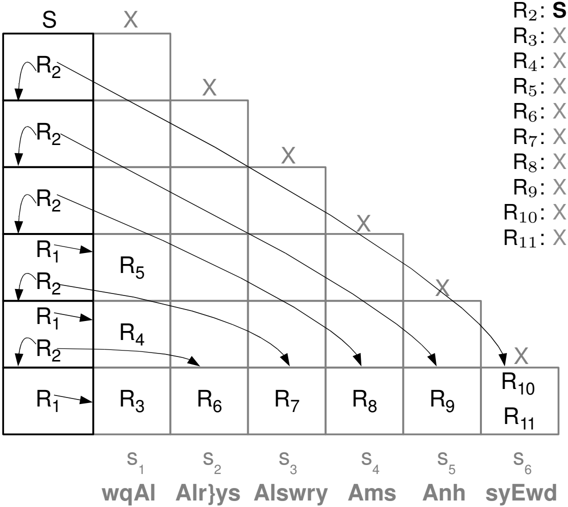
R_1: $S \rightarrow \langle X\ ,\ X \rangle$
R_2: $S \rightarrow \langle S\ X\ ,\ S\ X \rangle$
R_3: **$X \rightarrow \langle s_1$, said$\rangle$**
R_4: $X \rightarrow \langle s_1\ s_2$, the president said$\rangle$
R_5: $X \rightarrow \langle s_1\ s_2\ s_3$, Syrian president says$\rangle$
R_6: **$X \rightarrow \langle s_2$, president$\rangle$**
R_7: **$X \rightarrow \langle s_3$, the Syrian$\rangle$**
R_8: **$X \rightarrow \langle s_4$, yesterday$\rangle$**
R_9: **$X \rightarrow \langle s_5$, that$\rangle$**
R_{10}: **$X \rightarrow \langle s_6$, would return$\rangle$**
R_{11}: **$X \rightarrow \langle s_6$, he would return$\rangle$**

Keeping Track of All Derivations. CYK Grid



R_1: S→⟨X , X⟩
R_2: S→⟨S X , S X⟩
R_3: X→⟨s_1 , said⟩
R_4: X→⟨s_1 s_2 , the president said⟩
R_5: X→⟨s_1 s_2 s_3 , Syrian president says⟩
R_6: X→⟨s_2 , president⟩
R_7: X→⟨s_3 , the Syrian⟩
R_8: X→⟨s_4 , yesterday⟩
R_9: X→⟨s_5 , that⟩
R_{10}: X→⟨s_6 , would return⟩
R_{11}: X→⟨s_6 , he would return⟩

Keeping Track of All Derivations. CYK Grid



R_1: **S**$\rightarrow\langle$**X , X**$\rangle$

R_2: **S**$\rightarrow\langle$**S X , S X**$\rangle$

R_3: $X\rightarrow\langle s_1$, said$\rangle$

R_4: $X\rightarrow\langle s_1\ s_2$, the president said$\rangle$

R_5: $X\rightarrow\langle s_1\ s_2\ s_3$, Syrian president says$\rangle$

R_6: $X\rightarrow\langle s_2$, president$\rangle$

R_7: $X\rightarrow\langle s_3$, the Syrian$\rangle$

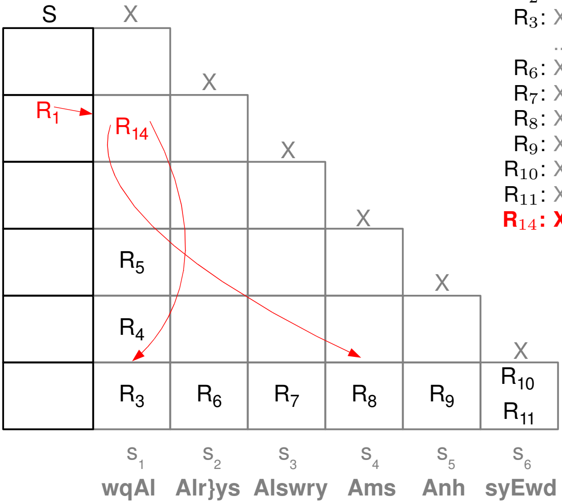
R_8: $X\rightarrow\langle s_4$, yesterday$\rangle$

R_9: $X\rightarrow\langle s_5$, that$\rangle$

R_{10}: $X\rightarrow\langle s_6$, would return$\rangle$

R_{11}: $X\rightarrow\langle s_6$, he would return$\rangle$

Keeping Track of All Derivations. CYK Grid (2)



R_1: $S \rightarrow \langle X\ ,\ X \rangle$

R_2: $S \rightarrow \langle S\ X\ ,\ S\ X \rangle$

R_3: $X \rightarrow \langle s_1\ ,\ \text{said} \rangle$

...

R_6: $X \rightarrow \langle s_2\ ,\ \text{president} \rangle$

R_7: $X \rightarrow \langle s_3\ ,\ \text{the Syrian} \rangle$

R_8: $X \rightarrow \langle s_4\ ,\ \text{yesterday} \rangle$

R_9: $X \rightarrow \langle s_5\ ,\ \text{that} \rangle$

R_{10}: $X \rightarrow \langle s_6\ ,\ \text{would return} \rangle$

R_{11}: $X \rightarrow \langle s_6\ ,\ \text{he would return} \rangle$

R_{14}: $X \rightarrow \langle X_1\ s_2\ X_2\ s_4\ s_5\ ,$
$\text{y'day}\ X_2\ \text{president}\ X_1\ \text{that} \rangle$

Hierarchical Phrase-Based Decoding

Given:

- A source sentence s
- A stochastic Synchronous Context Free Grammar (SCFG) G
- An n-gram Language Model M, represented as a WFSA

Decoding is done (ideally) in three steps:

1. Apply the translation grammar: $\mathcal{T} = \Pi_2(\{s\} \circ G)$
   - $\{s\}$ can be applied to $\mathcal{G}$ using the CYK algorithm, as described
2. Apply the language model via intersection: $\mathcal{L} = \mathcal{T} \cap M$,
3. Find the highest scoring path under both $\mathcal{G}$ and $\mathcal{M}$ (a.k.a. shortest distance): $\text{argmax}\ \mathcal{L}$

Representation chosen for $\mathcal{T}$ determine the form and complexity of the intersection and shortest path algorithms used in Steps 2 and 3

Hierarchical Phrase-Based Decoding Architectures

Different representations of $\mathcal{T}$ can lead to different decoder architectures

- Hypergraphs: Cube Pruning Decoder [4]
- FSAs as expansions of RTNs [5] [6]: **HiFST** [7]
- Push-Down Automata (PDAs) as replacements of RTNs: **HiPDT** [8] implemented in OpenFST [9] [10]

The space of translations $\mathcal{T} = \Pi_2(\{s\} \circ \mathcal{G})$ is well-characterized:

- $\mathcal{T}$ is a weighted context-free language
- If $\mathcal{G}$ does not allow unbounded insertions, $\mathcal{T}$ is a regular language
- Steps 2 and 3 in decoding can be done with FSM techniques

---

[4] D. Chiang. *Hierarchical phrase-based translation.* Computational Linguistics, 2007

[5] W. Woods. *Transition network grammars for natural language analysis*. Comm. ACM, 1970

[6] M. Mohri. *Finite-state transducers in language and speech processing*. Computational Linguistics, 1997

[7] A. de Gispert et al. *Hierarchical phrase-based translation with weighted finite state transducers and shallow-n grammars*. Computational Linguistics, 2010.

[8] G. Iglesias et al. *Hierarchical Phrase-based Translation Representations*. EMNLP 2011.

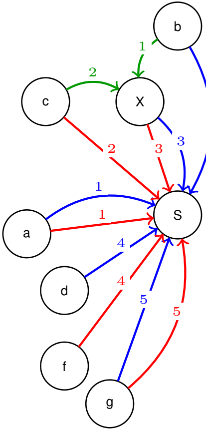
[9] C. Allauzen et al. *OpenFst: A General and Efficient Weighted Finite-State Transducer Library.* CIAA, 2007.

[10] C. Allauzen and Michael Riley. *Pushdown Transducers*. http://pdt.openfst.org

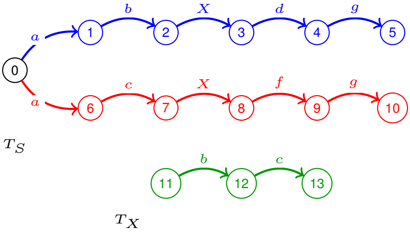
Alternative Representations: Hypergraphs and RTNs

A simple case (target-side only) : $S \rightarrow a\, b\, X\, d\, g$ $\quad S \rightarrow a\, c\, X\, f\, g$ $\quad X \rightarrow b\, c$

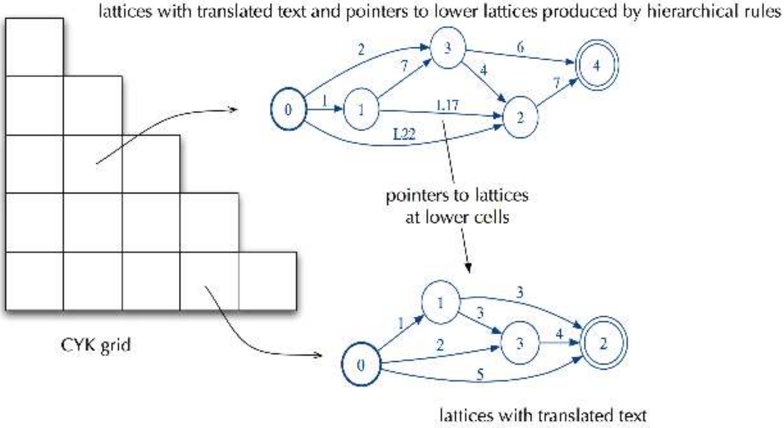
Hypergraph



RTN



RTN Construction – $\Pi_2(\{s\} \circ \mathcal{G})$



- Easy implementation with FST Replace operation
- Usual FST operations can be applied to skeleton $\rightarrow$ lattice size reduction

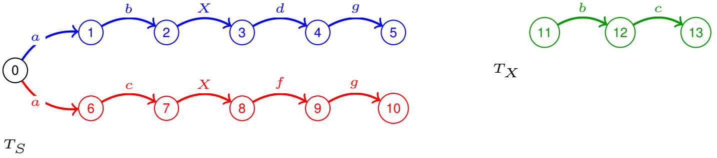
Output has the form of a Recursive Transition Network (RTN) [11] [12]

[11] Woods, W. A. 1970. Transition network grammars for natural language analysis. Commun. ACM

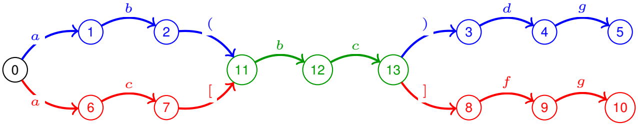
[12] Mohri, Mehryar. 1997. Finite-state transducers in language and speech processing. Computational Linguistics,.

Alternative Representations: PDAs, FSAs from RTNs

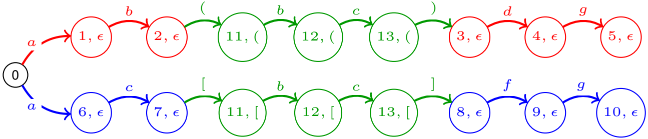
$S \rightarrow a\,b\,X\,d\,g$ $\quad S \rightarrow a\,c\,X\,f\,g$ $\quad X \rightarrow b\,c$



RTN representation of $\mathcal{T}$

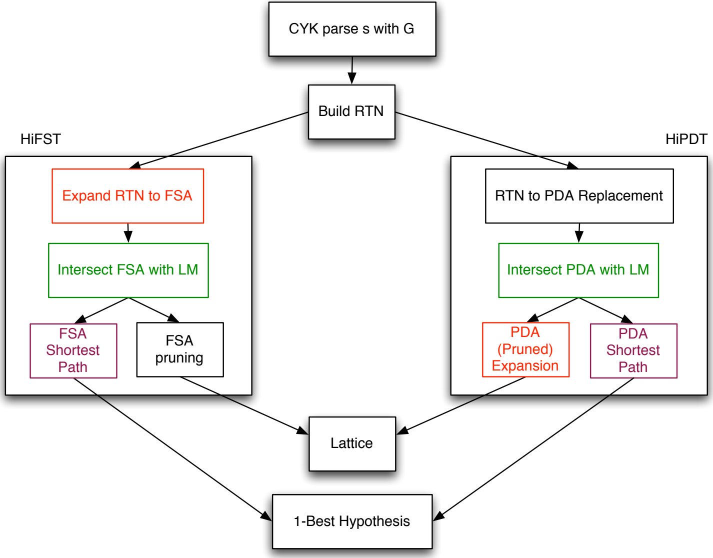


PDA equivalent to the RTN – Derived by the Replacement Algorithm



FSA equivalent to the PDA – Derived by the Expansion Algorithm

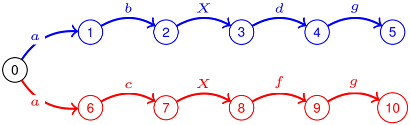
HiPDT and HiFST – Common Architecture



Optimised Translation Representations – RTN

RTN, PDA, and FSA can benefit from FSA epsilon removal, determinization and minimization algorithms applied to their components (for RTNs and PDAs) or their entirety (for FSAs).

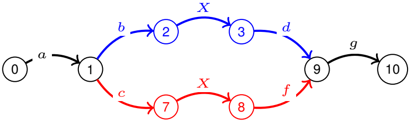
RTN



T_S

T_X

‘Optimised’ RTN



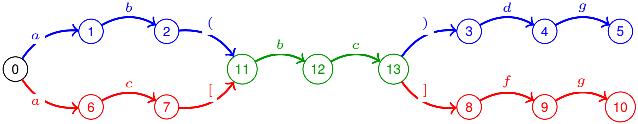
T_S

T_X

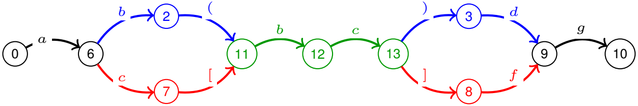
Optimised Translation Representations – PDA

RTN, PDA, and FSA can benefit from FSA epsilon removal, determinization and minimization algorithms applied to their components (for RTNs and PDAs) or their entirety (for FSAs).

PDA



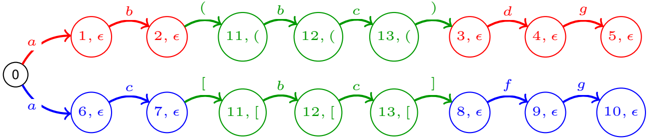
‘Optimised’ PDA



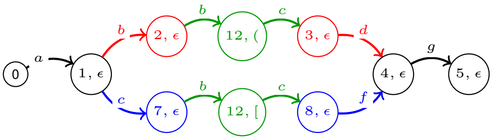
Optimised Translation Representations – FSA

RTN, PDA, and FSA can benefit from FSA epsilon removal, determinization and minimization algorithms applied to their components (for RTNs and PDAs) or their entirety (for FSAs).

FSA

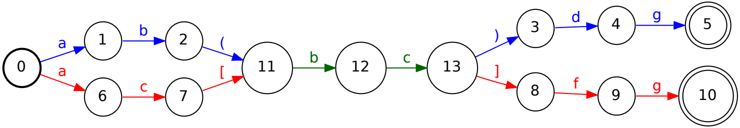


‘Optimised’ FSA



Push-Down Automata

- Informally: PDA is an FSA with a stack
- PDT extension [13] implemented in OpenFST [14].
  - 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

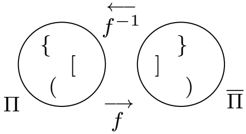


[13] C. Allauzen and Michael Riley. *Pushdown Transducers*. http://pdt.openfst.org

[14] C. Allauzen et al. *OpenFst: A General and Efficient Weighted Finite-State Transducer Library.* CIAA, 2007.

Dyck Language: balanced strings over parentheses

Let Π and $\overline{\Pi}$ be two finite alphabets with a bijection f



$$a \in \Pi \Rightarrow \bar{a} \in \overline{\Pi}$$
$$a \in \overline{\Pi} \Rightarrow \bar{a} \in \Pi$$

The Dyck language D_Π over $\widehat{\Pi} = \Pi \cup \overline{\Pi}$ is defined by

$$S \to \epsilon$$
$$S \to S\,S$$
$$S \to a\,S\,\bar{a} \;\; \forall a \in \Pi$$

1. Define a mapping $c_\Pi : \widehat{\Pi}^* \to \Pi^*$.
$c_\Pi(x)$ is the string derived from x by iterative deletion of all pairs $a\,\bar{a}$ for $a \in \Pi$.

   $\Rightarrow$ If $x \in D_\Pi$ then $c_\Pi(x) = \epsilon$. Similarly, $D_\Pi = c_\Pi^{-1}(\epsilon)$.

2. For 2 finite sets A and B, $B \subset A$, define $r_B : A^* \to B^*$ by

$$r_B(x_1 \dots x_n) = y_1 \dots y_n \text{ where } y_n = \begin{cases} x_n & \text{if } x_i \in B \\ \epsilon & \text{if } x_i \notin B \end{cases}$$

   $\Rightarrow$ r_B is a filter that erases symbols not in B

Push-Down Automata – Definitions

A weighted pushdown automaton (PDA) T over the tropical semiring $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ is a 9-tuple $(\Sigma, \Pi, \overline{\Pi}, Q, E, I, F, \rho)$ where

- Σ is the finite input alphabet
- Π and $\overline{\Pi}$ are the finite open- and close-parentheses alphabets
- Q is a finite set of states
- $I \in Q$ the initial state
- $F \subseteq Q$ the set of final states
- $E \subseteq Q \times (\Sigma \cup \widehat{\Pi} \cup \{\epsilon\}) \times (\mathbb{R} \cup \{\infty\}) \times Q$ a finite set of transitions
  Transitions in E are denoted $e = (p[e], i[e], w[e], n[e])$.
- $\rho : F \to \mathbb{R} \cup \{\infty\}$ the final weight function.

PDA Paths

A path π is a sequence of transitions $\pi = e_1 \dots e_n$ s.t. $n[e_i] = p[e_{i+1}]$

- $i[\pi] = i[e_1] \dots i[e_n]$ – input symbols
- $w[\pi] = w[e_1] + \dots + w[e_n]$ – path weight
- A path is accepting if $p[\pi] = I$ and $n[\pi] \in F$
- π is balanced if $r_{\widehat{\Pi}}(i[\pi]) \in D_\Pi$
- A balanced path accepts the string $x \in \Sigma^*$ if $r_\Sigma(i[\pi]) = x$

The weight associated by T to x is

$$T(x) = \min_{\pi \in P(x)} w[\pi] + \rho(n[\pi])$$

where $P(x)$ is the set of balanced paths accepting x

Bounded Stacks

A PDA T has a bounded stack if $\exists K \in \mathbb{N}$ such that

$$|\, c_\Pi(r_{\hat{\Pi}}(i[\pi]))\,| \leq K$$

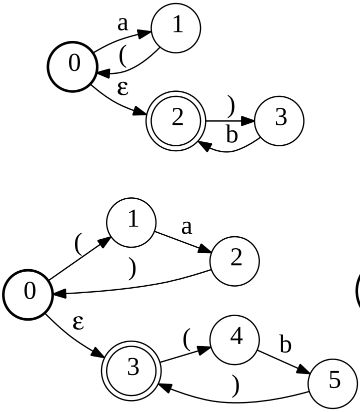
for any sub-path π of any balanced path in T

An example:

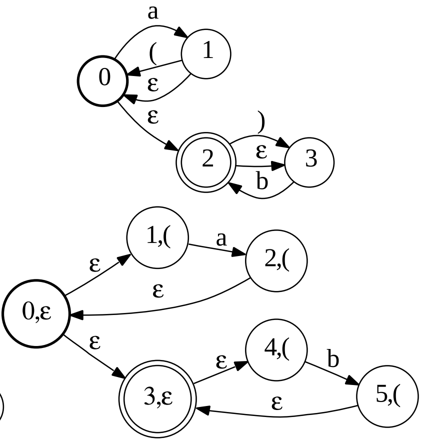
$$\begin{array}{ccccccccc} i[\pi] & r_{\hat{\Pi}} & r_{\hat{\Pi}}(i[\pi]) & c_\Pi & c_\Pi(r_{\hat{\Pi}}(i[\pi]) & & |c_\Pi(r_{\hat{\Pi}}(i[\pi])| \\ B(A[B(AB) & \Rightarrow & ([() & \Rightarrow & ([& \Rightarrow & 2 \end{array}$$

Finite number of unmatched open parentheses $\Leftrightarrow$ bounded-stack

PDA Examples

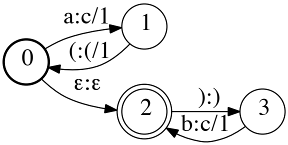


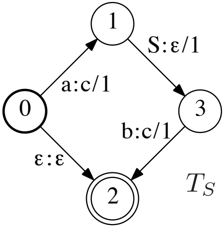
(t) Non-regular PDA accepting $\{a^n b^n | n \in \mathbb{N}\}$.
(b) Bounded-stack PDA accepting a^*b^* and



(t) Regular (not bounded-stack) PDA accepting a^*b^*.
(b) its expansion as an FSA.

PDT Examples

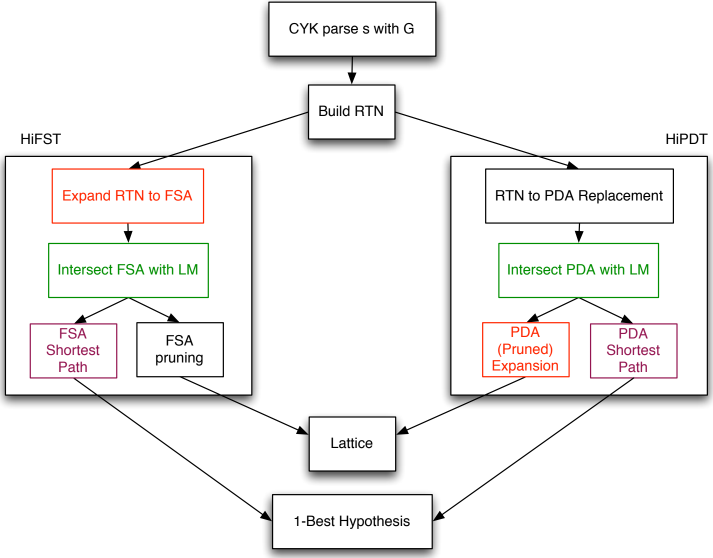




Weighted PDT representing $(a^n b^n, c^{2n})$

Equivalent RTN

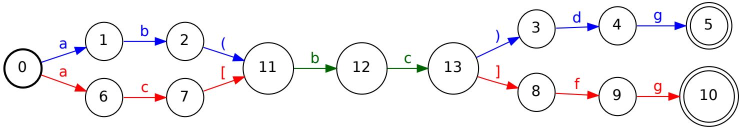
HiPDT – PDA (Pruned) Expansion



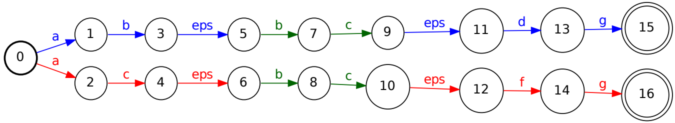
Expansion of PDAs to FSAs

A bounded stack PDA can be expanded into an equivalent FSA

- Bounded-stack PDA:

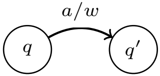
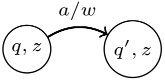
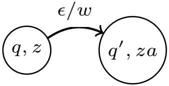
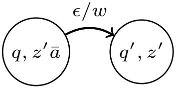


- Its expansion as an FSA:

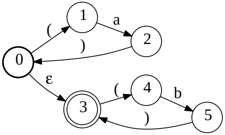


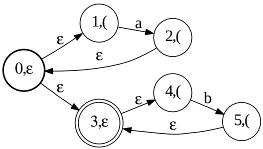
- Change in topology; no parentheses

PDA Expansion Algorithm

PDA T	FSA expansion T'	Input Symbol in T
		$a \in \Sigma \cup \{\epsilon\}$
		$a \in \Pi$
		$a \in \overline{\Pi}$

Initial state: $I' = (I, \epsilon)$ Final states: $(q, \epsilon), q \in F$





PDA Pruned Expansion

Input: a bounded-stack PDA T, and a pruning threshold β

Output: a pruned FST T'_β such that

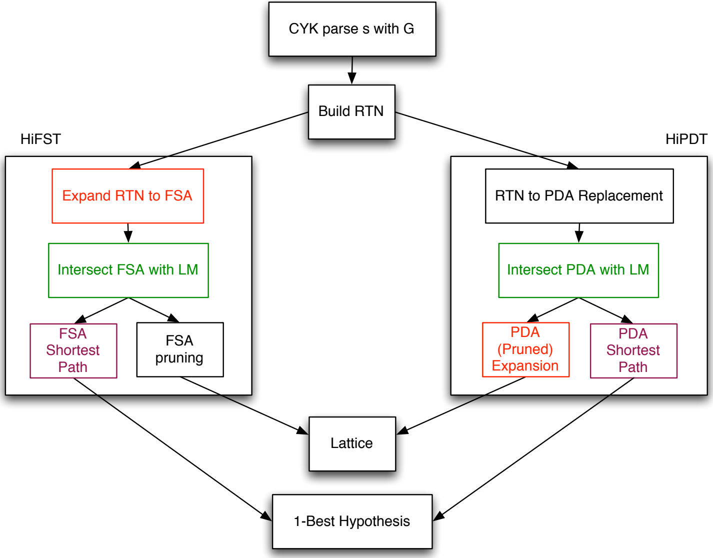
- States and transitions are deleted if there is no accepting path π in T' such that

$$\lambda'(p[\pi]) + w[\pi] + \rho'(n[\pi]) \leq d + \beta$$

  where d is the shortest distance in T.

- Equivalent to expansion of PDA T to an FSA T' followed by pruning

HiPDT – Intersection PDA with LM



Intersection of PDAs with FSAs

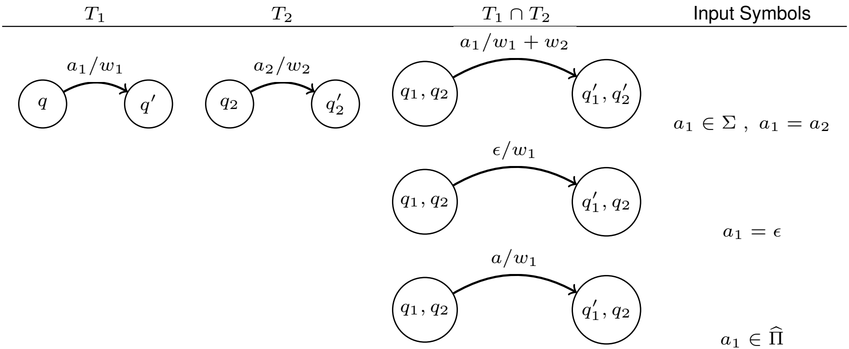
- PDA T intersection with FSA M is closed (Bar-Hillel intersection)
- Almost identical to FSA intersection
  - parentheses treated as epsilons but retained as parentheses in the result
- Time/Space complexity: $O(|T||M|)$

Intersecting a PDA T_1 with an FSA T_2 :

$$T_1 \ , \ T_2 \ \ : (T_1 \cap T_2)(x) = T_1(x) + T_2(x)$$

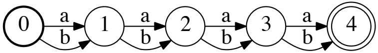
Intersecting a PDA T_1 with an FSA T_2

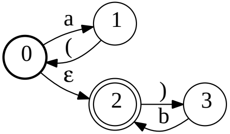
T_1 , T_2 : $(T_1 \cap T_2)(x) = T_1(x) + T_2(x)$ (asssuming T_2 has no epsilons) :

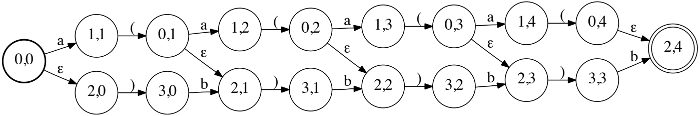


Initial: $I = (I_1, I_2)$ Final: (q_1, q_2) $q_1 \in F_1$, $q_2 \in F_2$ $(\rho(q_1, q_2) = \rho_1(q_1) + \rho_2(q_2))$

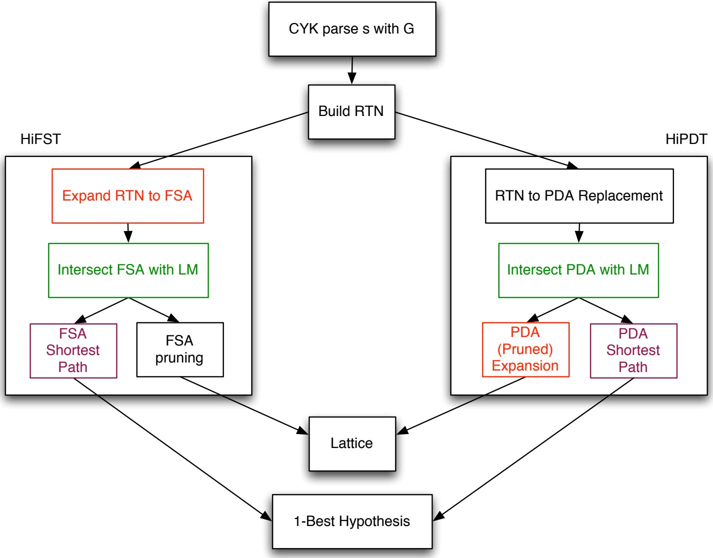
Intersection of FSA accepting $\{a,b\}^4$ and PDA accepting $\{a^n, b^n\}$







HiPDT – Shortest Path / Distance



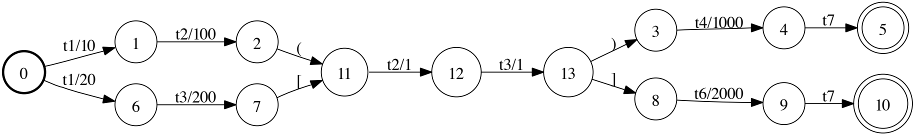
Shortest Distance Algorithm

ShortestDistance(T)
for each $q \in Q$ and $a \in \Pi$ **do**
  $B[q, a] \leftarrow \emptyset$
GetDistance(T, I)
return $d[f, I]$

Relax$(q, s, w, \mathcal{S})$
if $d[q, s] > w$ **then**
  $d[q, s] \leftarrow w$
  if $q \notin \mathcal{S}$ **then**
    Enqueue$(\mathcal{S}, q)$

GetDistance(T, s)
for each $q \in Q$ **do**
  $d[q, s] \leftarrow \infty$
$d[s, s] \leftarrow 0$
$\mathcal{S}_s \leftarrow s$
while $\mathcal{S}_s \neq \emptyset$ **do**
  $q \leftarrow$ Head$(\mathcal{S}_s)$
  Dequeue$(\mathcal{S}_s)$
  for each $e \in E[q]$ **do**
    if $i[e] \in \Sigma \cup \{\epsilon\}$ **then**
      Relax$(n[e], s, d[q, s] + w[e], \mathcal{S}_s)$
    elseif $i[e] \in \overline{\Pi}$ **then**
      $B[s, \overline{i[e]}] \leftarrow B[s, \overline{i[e]}] \cup \{e\}$
    elseif $i[e] \in \Pi$ **then**
      if $d[n[e], n[e]]$ is undefined **then**
        GetDistance$(T, n[e])$
      for each $e' \in B[n[e], i[e]]$ **do**
        $w \leftarrow d[q, s] + w[e] + d[p[e'], n[e]] + w[e']$
        Relax$(n[e'], s, w, \mathcal{S}_s)$

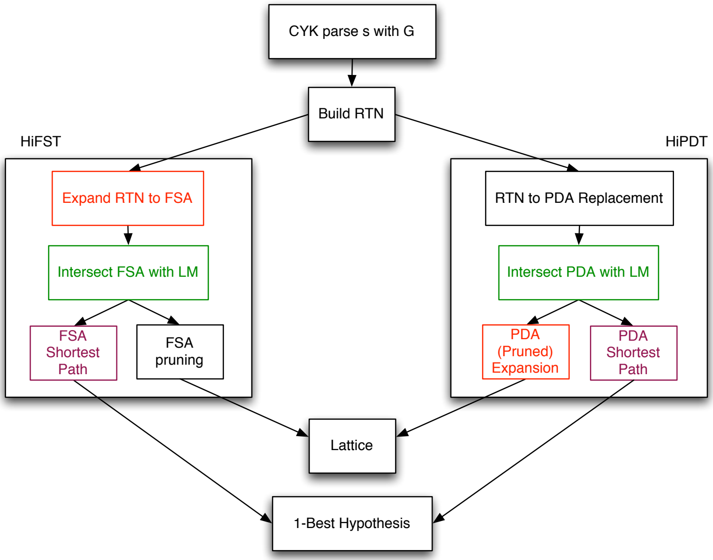
Shortest Distance



s_1	s_2	$d[s_1, s_2]$	$B[s_1, \bar{i}[e]]$
0	0	0	-
0	1	10	-
0	2	110	-
11	**11**	**0**	-
11	**12**	**1**	-
11	**13**	**2**	-
11	**3**	-	**(13,),0,3)**
11	**8**	-	**(13,],0,8)**
0	3	112	-
...			
0	6	20	-
0	7	220	-
0	8	222	-
...			

- Memoization of shortest distances
- Complexity
  - General PDA: $O(|T|^3)$
  - PDA derived from acyclic RTN: $O(|T|)$
  - $\Rightarrow$ Same PDA intersected with a finite M: $O(|T||M|^2)$

HiPDT – RTNs to PDAs



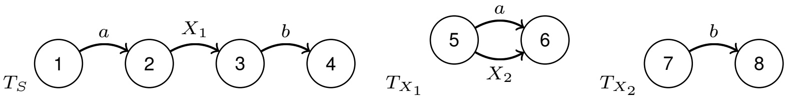
RTN Definitions

An RTN R : $(\mathbb{N}, \Sigma, (T_\nu)_{\nu \in \mathbb{N}}, S)$ is defined as

- $\mathbb{N}$ – alphabet of non-terminals
- $(T_\nu)_{\nu \in \mathbb{N}}$ – a family of FSAs with input alphabet $\Sigma \cup \mathbb{N}$
  - T_S is the root FSA
- $S \in \mathbb{N}$ – root non-terminal

A string $x \in \Sigma^*$ is accepted by R if there is an accepting path in T_S such that recursively replacing every transition with the label $\nu \in \mathbb{N}$ by a path from T_ν leads to a path π^* such that $x = i[\pi^*]$.

R : $\Sigma = \{a, b\}$, $\mathbb{N} = \{X_1, X_2\}$

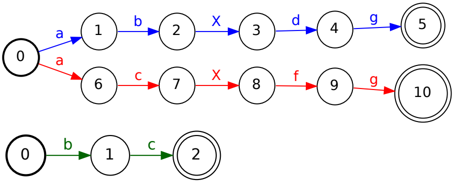


R accepts aab and abb

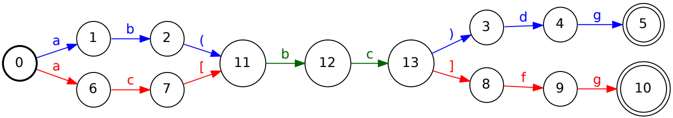
Replacement transforms a Recursive Transition Network into a PDA

- RTN:

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



- PDA:



- The RTN and the PDA are equivalent.
- For our applications, the RTNs have finite recursion levels, ensuring that the PDAs have bounded stack.

Replacement Algorithm for RTNs

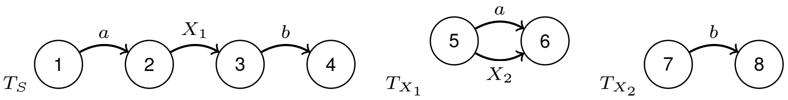
RTN $R \Rightarrow$ PDA T (for simplicity, each RTN FSA has a single final state F_ν)

$T : (\Sigma, \Pi, \overline{\Pi}, Q, E, I_S, F_S, \rho_S)$

$$\Pi = Q = \cup_{\nu \in \mathbb{N}} Q_\nu \qquad E = \cup_{\nu \in \mathbb{N}} \cup_{e \in E_\nu} \{ (p[e], n[e], w[e], I_\nu) \ , \ (F_\nu, \overline{n[e]}, \rho[e], n[e]) \}$$

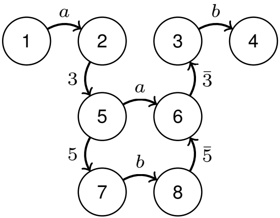
RTN R

R accepts aab and abb



PDT T

T accepts $a3a\bar{3}b$ and $a35b\bar{5}\bar{3}b$



Hierarchical Phrase-Based Translation

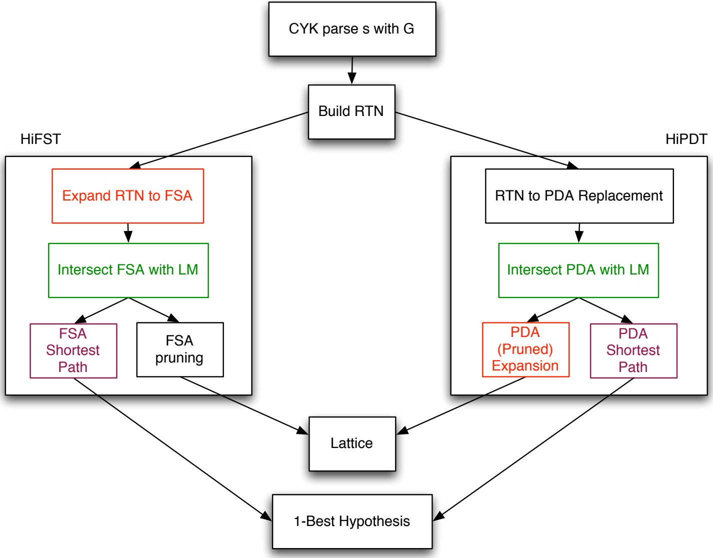
Recall the SMT problem
Given:

- A source sentence s
- A stochastic Synchronous Context Free Grammar (SCFG) G
- An n-gram Language Model M, represented as a WFSA

Decoding is done (ideally) in three steps:

1. Apply the translation grammar: $\mathcal{T} = \Pi_2(\{s\} \circ G)$
2. Apply the language model: $\mathcal{L} = \mathcal{T} \cap M$,
3. Find the highest scoring path under both $\mathcal{G}$ and $\mathcal{M}$: $\mathrm{argmax}\ \mathcal{L}$

HiPDT and HiFST – Common Architecture



Complexities of Hiero Decoders

Translation complexity of target language representations for translation grammars of rank 2.

Representation	Time Complexity	Space Complexity
CFG/hypergraph	$O(\|s\|^3 \|G\| \|M\|^3)$	$O(\|s\|^3 \|G\| \|M\|^3)$
PDA [15] [16]	$O(\|s\|^3 \|G\| \|M\|^3)$	$O(\|s\|^3 \|G\| \|M\|^2)$
FSA[17]	$O(e^{\|s\|^3 \|G\|} \|M\|)$	$O(e^{\|s\|^3 \|G\|} \|M\|)$

- HiPDT will be more efficient than HiFST for large grammars, if language model is small
- HiFST more efficient with bigger language models and smaller grammar

This is all worst-case: HiFST and HiPDT are faster in practice due to optimizations over RTN

- For example, in translation of a 15 word sentence, expansion of an RTN yields a WFSA with 174×10^6 states.
- If the RTN is determinised and minimised prior to expansion, the resulting WFSA has only 34×10^3 states.

[15] G. Iglesias et al. *Hierarchical Phrase-based Translation Representations*. EMNLP 2011

[16] C. Allauzen et al. *Pushdown Automata in Statistical Machine Translation*, under review at Computational Linguistics

[17] A. de Gispert et al. *Hierarchical Phrase-based Translation with Weighted Finite State Transducers and Shallow-N Grammars.* Computational Linguistics, 2010

Complexity for Non-Hiero Grammars

- In general, a hypergraph can be exponentially larger than a corresponding optimized PDT, but a PDT can represent any hypergraph in linear space.
- SCFGs of arbitrary rank l_N

Representation	Time Complexity
Hypergraphs	$O(\lvert G\rvert\lvert s\rvert^{l_N+1}\lvert M\rvert^{l_N+1})$
PDAs	$O(\lvert G\rvert\lvert s\rvert^{l_N+1}\lvert M\rvert^{3})$
FSAs	$O(e^{\lvert G\rvert\lvert s\rvert^{l_N+1}}\lvert M\rvert)$

- PDAs might be useful for more complex grammars, such as SAMT[18], or GHKM[19]

[18] Zollmann, A., A. Venugopal. *Syntax augmented machine translation via chart parsing*. WMT'2006

[19] Galley, M. et al. *What's in a translation rule?* HLT'2004

Challenge: Develop a decoding strategy for HiPDT

Complexity analysis suggests that HiPDT prefers

- large translation grammars G
- small(er) language models M

Strategy: Rescoring based on entropy-pruned n-gram language models[20]

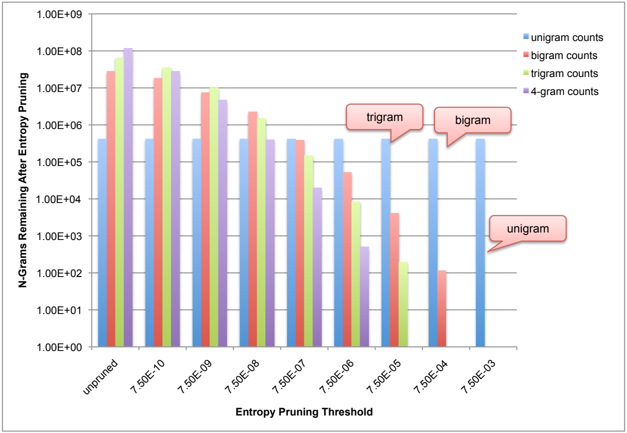
- Successfully used in speech recognition systems[21]
- Not widely used in SMT

[20] A. Stolcke. *Entropy-based Pruning of Backoff Language Models.* DARPA Broadcast News Transcription and Understanding Workshop,1998.

[21] Andrej Ljolje et al. *Efficient general lattice generation and rescoring.* Eurospeech,1999.

Entropy Pruning of N-Gram Language Models

Entropy pruning can be used to reduce the complexity of n-gram language models

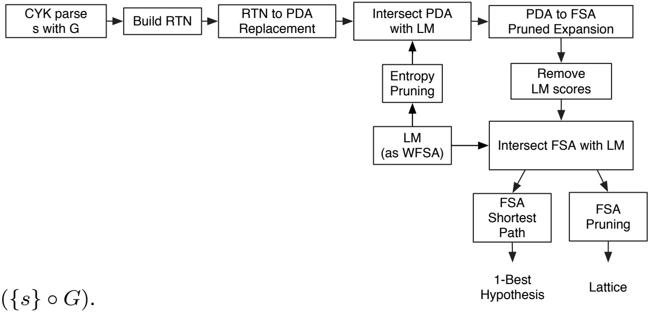


Entropy Pruning of First-Pass 4-Gram Language Model M_1

Decoding Pipeline with Entropy-Pruned LMs

Models

- Hierarchical Grammar G and pruned Language Model M^θ for decoding
- Large Language Model M for rescoring



Pipeline

1. $\mathcal{T} = \Pi_2(\{s\} \circ G)$.
2. Prune $\mathcal{T} \cap M^\theta$ at beam-width β
3. Remove M^θ scores from FSA
4. Rescore with M
5. Further rescoring operations, e.g. rescoring with much larger LM M_2...

For each θ, there will be no search errors in step 2 if β is large enough.
This approach requires the decoder to generate large/dense output FSAs.

Efficient Removal of LM Scores Using Lexicographic Semirings [22]

Two-dimensional weight

- weights are operated on indepdently
- second term interacts with first term only for tie-breaking

$$\langle w_1, w_2 \rangle \oplus \langle w_3, w_4 \rangle = \begin{cases} \langle w_1, w_2 \rangle & \text{if } w_1 < w_3 \text{ or } (w_1 = w_3 \text{ and } w_2 < w_4) \\ \langle w_3, w_4 \rangle & \text{otherwise} \end{cases}$$

$$\langle w_1, w_2 \rangle \otimes \langle w_3, w_4 \rangle = \langle w_1 + w_3, w_2 + w_4 \rangle$$

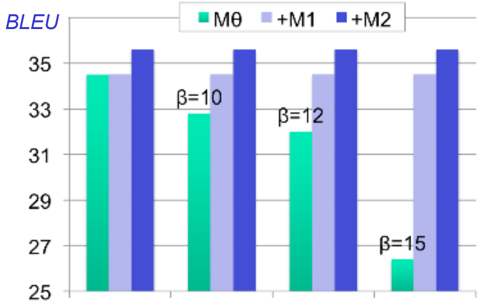
- In first pass decoding:
  - first dimension accumulates the first-pass LM and translation score, as usual
  - second dimension accumulates only the translation score
- Pruning is done under lexicographic semiring
  - Apart from ties, pruning is w.r.t. first-pass LM and translation scores in the first dimension
  - Translation scores are ‘carried along’ in the second dimension
- Scores in the first dimension are discarded after pruning

[22] B. Roark et al. *Lexicographic Semirings for Exact Automata Encoding of Sequence Models*. ACL 2011

Zh→En Translation with Compact Grammars

- Compact translation grammar
  - 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 β required
- Decoding speed-up



Entropy threshold θ	-	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

- Translate with very large translation grammar
- N-gram LM size controlled through entropy pruning

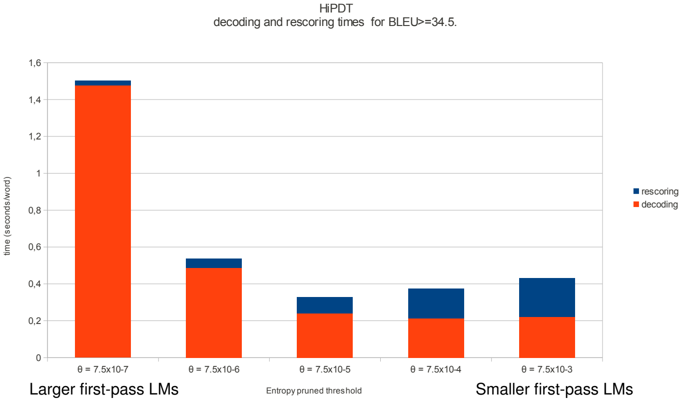
	HiFST			HiPDT		
entropy pruning θ	Success	Expand Fails	Intersect Fails	Success	Intersect Fails	Expand Fails
10^{-9}	12%	51%	37%	40%	8%	52%
10^{-8}	16%	53%	31%	76%	1%	23%
10^{-7}	18%	53%	29%	**99.8%**	0%	0.2%

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

Crucial to balance time spent in first-pass and second-pass operations

With more aggressive entropy pruning of the first-pass LM:

- Time spent in the first pass decreases because the first-pass LM is smaller
- But WFSAs produced from the first-pass are larger because the first-pass LM is weaker
- And so time spent in the second-pass increases



time spent generating FSAs from PDTs under first-pass LM

time spent rescoring FSAs with full LM

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[23] – complexity is hidden to the developers
- Not discussed:
  - Alignment under ITG grammars
  - Other LM smoothing strategies[24]
  - Work not published yet: weighted PDTs are proving useful in other large-scale NLP tasks

[23] See www.openfst.org

[24] Chelba et al. *Study on Interaction between Entropy Pruning and Kneser-Ney Smoothing.* Interspeech 2010.