Structured Reordering for Modeling Latent Alignments in Sequence Transduction

Bailin Wang[†], Mirella Lapata[†] and Ivan Titov[†§]

[†] ILCC, University of Edinburgh, [§] ILLC, University of Amsterdam

Systematic Generalization

Training Examples

what is the length of the colorado river ?

len(river(riverid ('colorado')))

what is the longest river ?

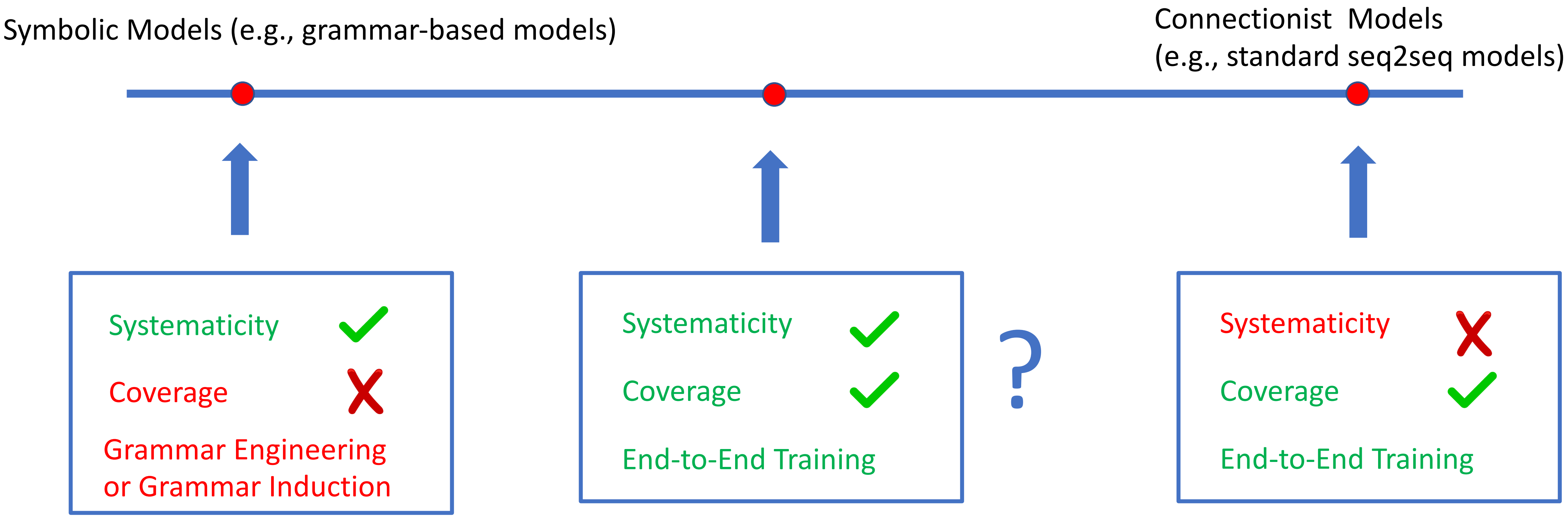
longest(river(all)))

Test Example

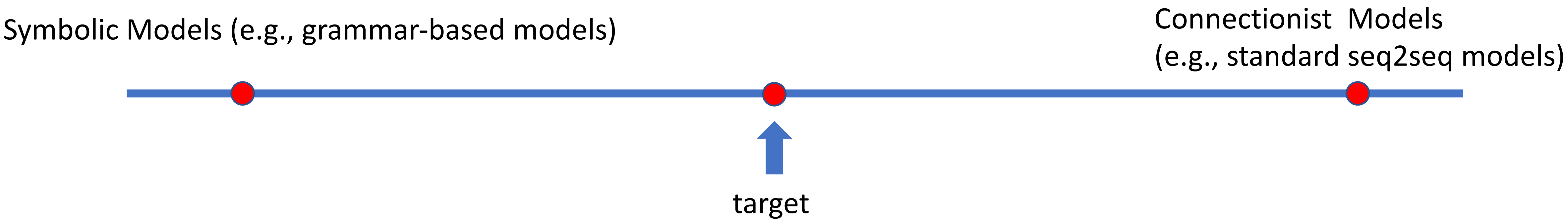
what is the length of the longest river ?

len(longest(river(all)))

The Spectrum of Sequence Transduction Models



The Spectrum of Sequence Transduction Models



Two inter-related questions:

1) Why are symbolic models good at systematic generalization?
2) What prevents seq2seq models from generalizing systematically?

Q1: Why are symbolic models good?

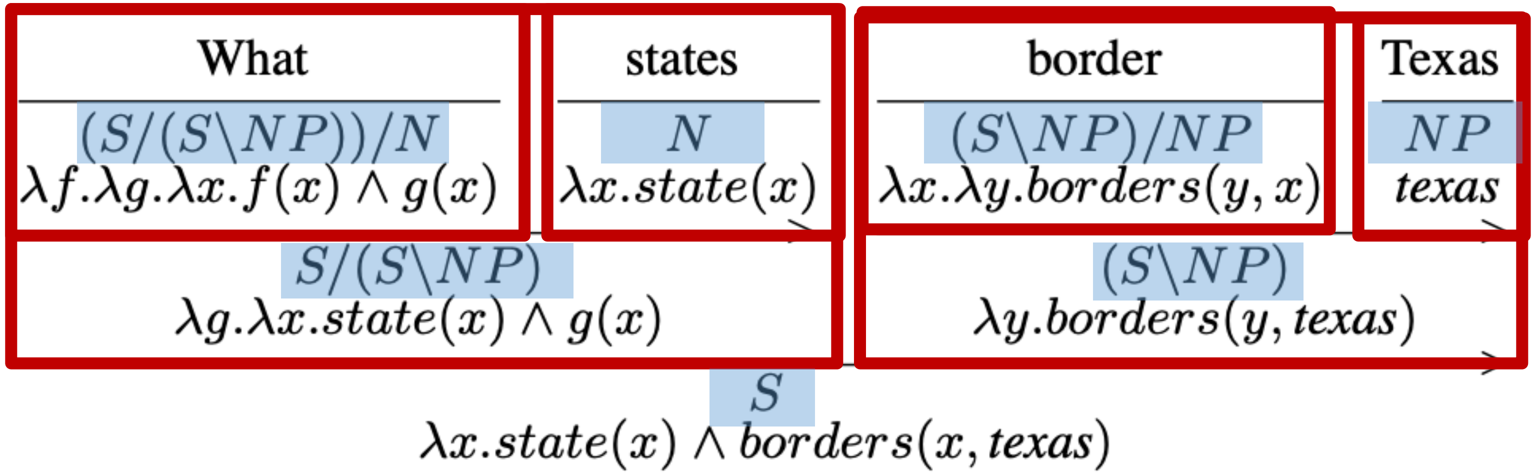


Figure: A CCG parse of the utterance "what states border Texas"

[*Luke S. Zettlemoyer and Michael Collins, Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars, 2012*]

Q1: Why are symbolic models good?

$$\frac{\text{What}}{\begin{array}{c}(S/(S\backslash NP))/N \\ \lambda f.\lambda g.\lambda x.f(x) \wedge g(x)\end{array}}$$

A CCG rule

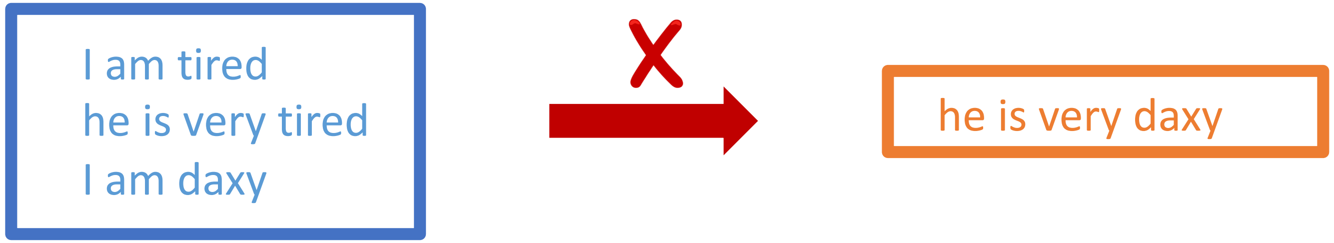
Their grammar rules implicitly encode alignments between *input and output segments*.

⬇

- *Explicit decomposition of input and output into segments*
- *consistent mappings from input segments to output ones*

Q2: What makes seq2seq fail?

- Primitive units (e.g., words) are inconsistently mapped across different contexts [1].

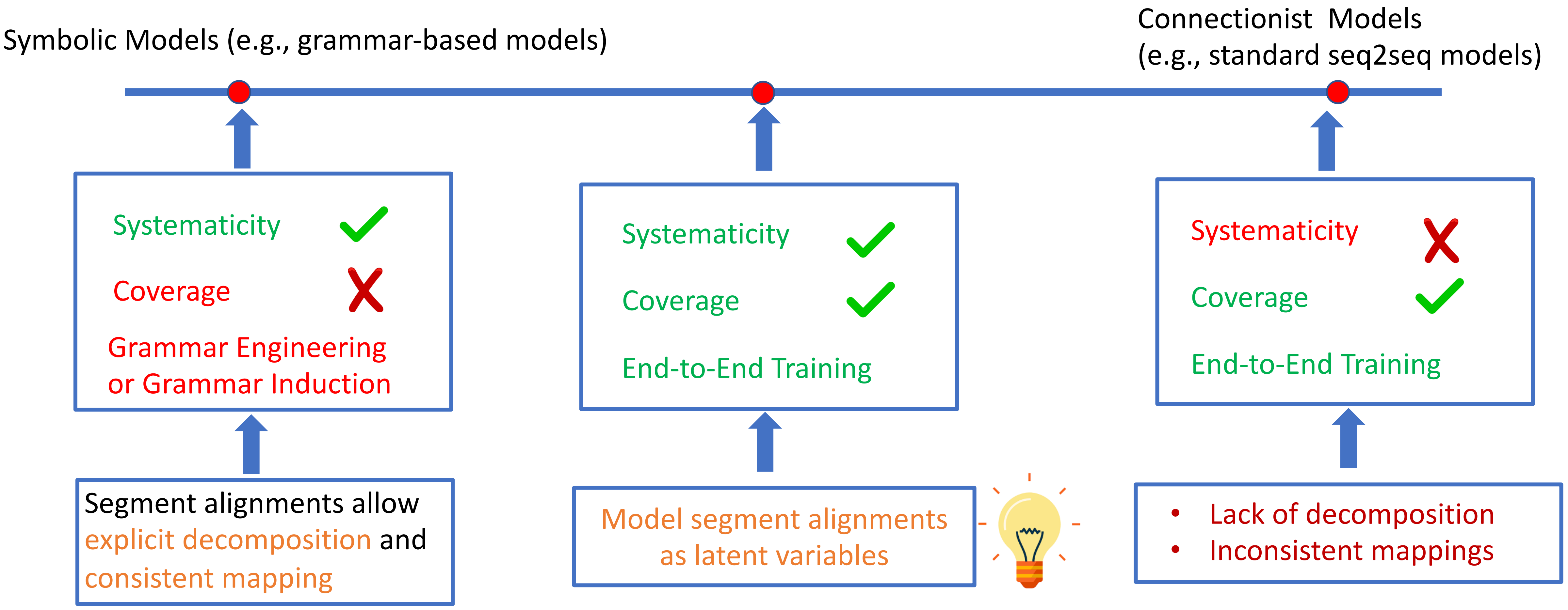


- Standard seq2seq models tend to memorize large chunks, e.g., they process ‘I am daxy’ as a whole [2].

They do not exhibit a strategy of decomposition and consistent mapping!

1. [*Brenden Lake, Marco Baroni, Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks, 2017*]
2. [*Dieuwke Hupkes, Verna Dankers, Mathijs Mul, Elia Bruni, Compositionality decomposed: how do neural networks generalise? 2019*]

Alignments for Systematicity



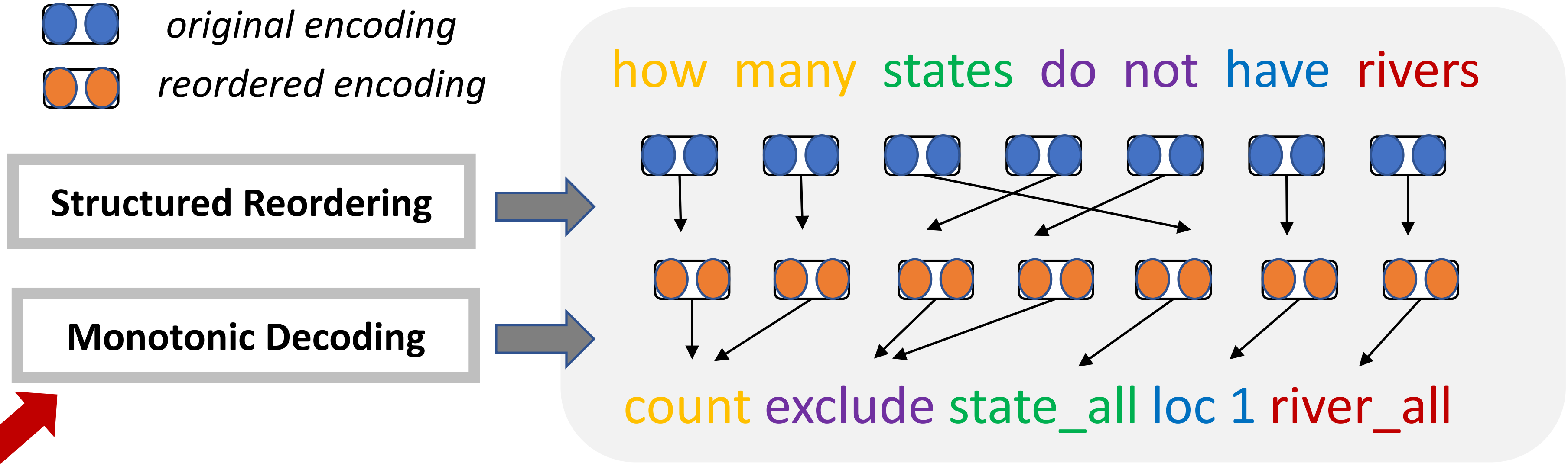
A seq2seq model endowed with *latent segment alignments*

Model Architecture *ReMoto*

Reordered-then-Monotone alignments

Discrete latent variables rather than attention

focus of this work



efficient modules in existing work [1,2]

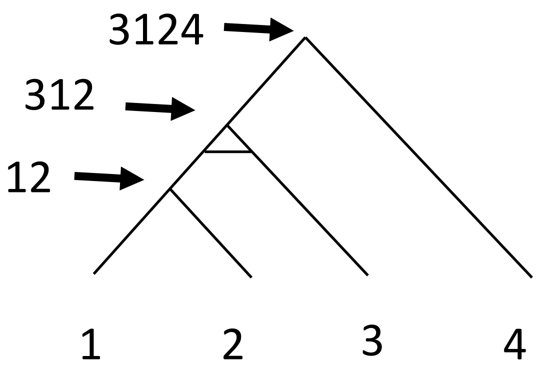
1. [Lei Yu, Jan Buys, and Phil Blunsom. Online segment to segment neural transduction, 2016]
2. [Chong Wang, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. Sequence modeling via segmentations. 2017]

Structured reordering via separable permutations

Permutation: 3 1 2 4

these five young lads → *young these five lads*

Separable Permutation:



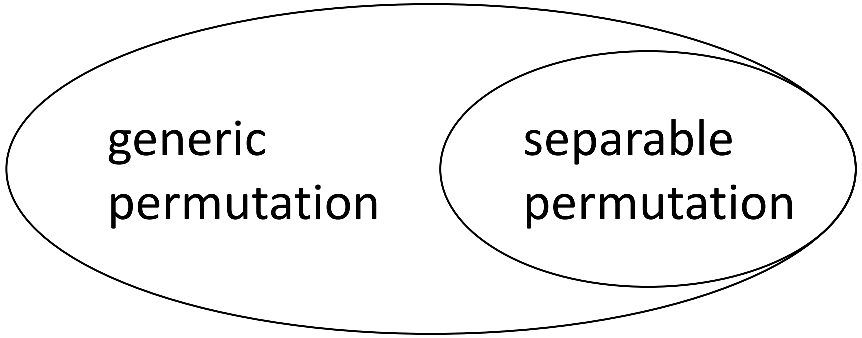
BTG rules
Λ : straight
△ : inverted

↔ 3 1 2 4

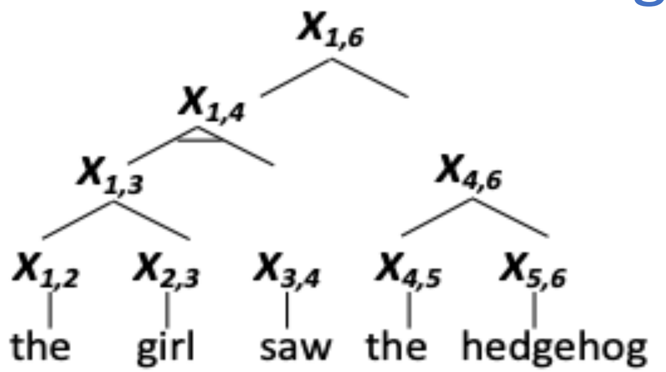
Forbidden patterns! 3 1 4 2 2 4 1 3

Why separable permutation?

Computational efficiency



Hierarchical modeling



Linguistic inductive bias [1,2]

1. [*Mark Steedman. A formal universal of natural language grammar, 2020*]
2. [*Miloš Stanojevic and Mark Steedman. Formal basis of a language universal, 2021*]

Training Objective:

reordered representations

$$-\log \underbrace{\mathbb{E}_{p_\phi(D|x)}}_{\text{parser for reordering}} \underbrace{p_\theta(y|\underline{M^D X})}_{\text{monotonic decoding}}$$

x, y	input and output sequence
X	encodings of input x
D	permutation tree
M^D	the permutation matrix corresponding to D

Surrogate Objective:

$$-\log p_\theta(y|M' X)$$

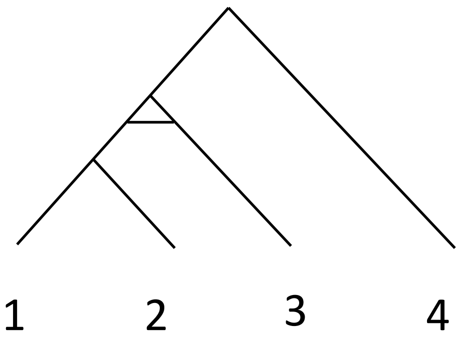
where

$$\underbrace{M' = \mathbb{E}_{p_\phi(D|x)} M^D}_{\text{expectation of permutation matrices}} \quad \text{or} \quad \underbrace{M' = \text{Perturb-and-MAP}[p_\phi(D|x)]}_{\text{an approximated sample of permutation matrix}}$$

Marginal Inference

MAP Inference

Constructing Permutation Matrices



Permutation Tree

3 1 2 4

0	0	1	0
1	0	0	0
0	1	0	0
0	0	0	1

Permutation Matrix

$$((M_1^2 \oplus M_2^3) \ominus M_3^4 \oplus M_4^5)$$

$$\boldsymbol{M_i^{i+1} = 1} \qquad \boldsymbol{A \oplus B = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}} \qquad \boldsymbol{A \ominus B = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix}}$$

Recursion underlying Marginal Inference

expected permutation matrix
for the segment from i to k

$$\underbrace{E_i^k}_{} = \underbrace{\sum_{i<j<k}}_{} \underbrace{p(straight)(E_i^j \oplus E_j^k) + p(inverted)(E_i^j \ominus E_j^k)}_{}$$

enumerate all middle points

for every middle point, there are two ways to combine
the permutation matrices from the left and right segments

Surrogate Objective: $-\log p_\theta(y|M'X)$

$$\underbrace{M' = E_1^n}_{\text{soft-ReMoto}} \qquad \underbrace{M' = \underbrace{\widetilde{E}_1^n}_{}}_{\text{hard-ReMoto}}$$

a perturbed variant based on
Straight-through Gumbel

Experiments

Diagnostic Task: Infix-to-Postfix Conversion

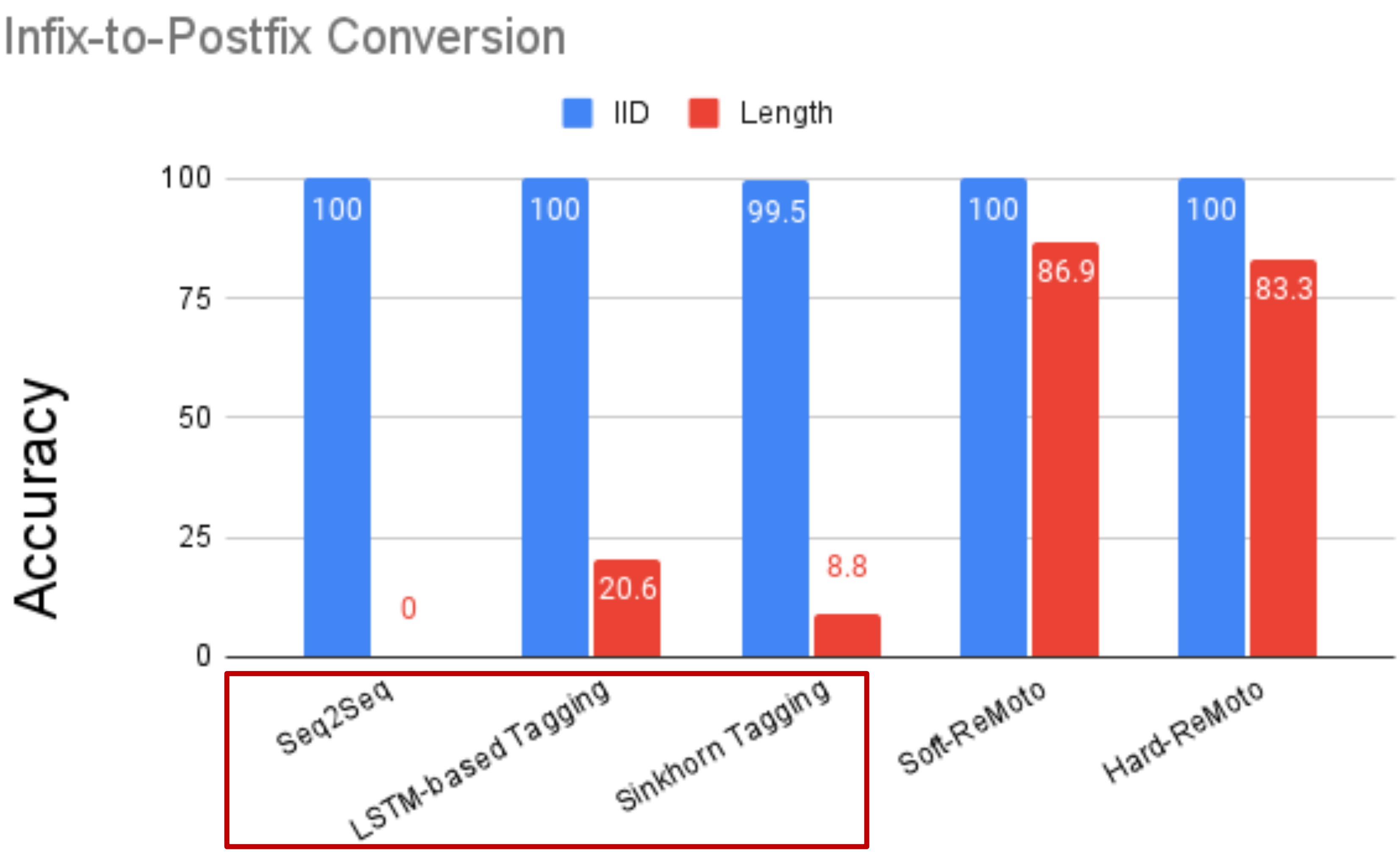
- Train (nesting depth < 7)

Input: $((1 + 9) * ((7 + 8) / 4))$ *Output:* $((1\ 9\ +)((7\ 8\ +)\ 4\ /)\ *)$

Input: $((6 + 5) * (3 + 2))$ *Output:* $((6\ 5\ +)(3\ 2\ +)\ *)$

- IID Evaluation (nesting depth < 7)
- Length Evaluation (nesting depth =7)

Results



Sinkhorn Network: [Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks, 2018]

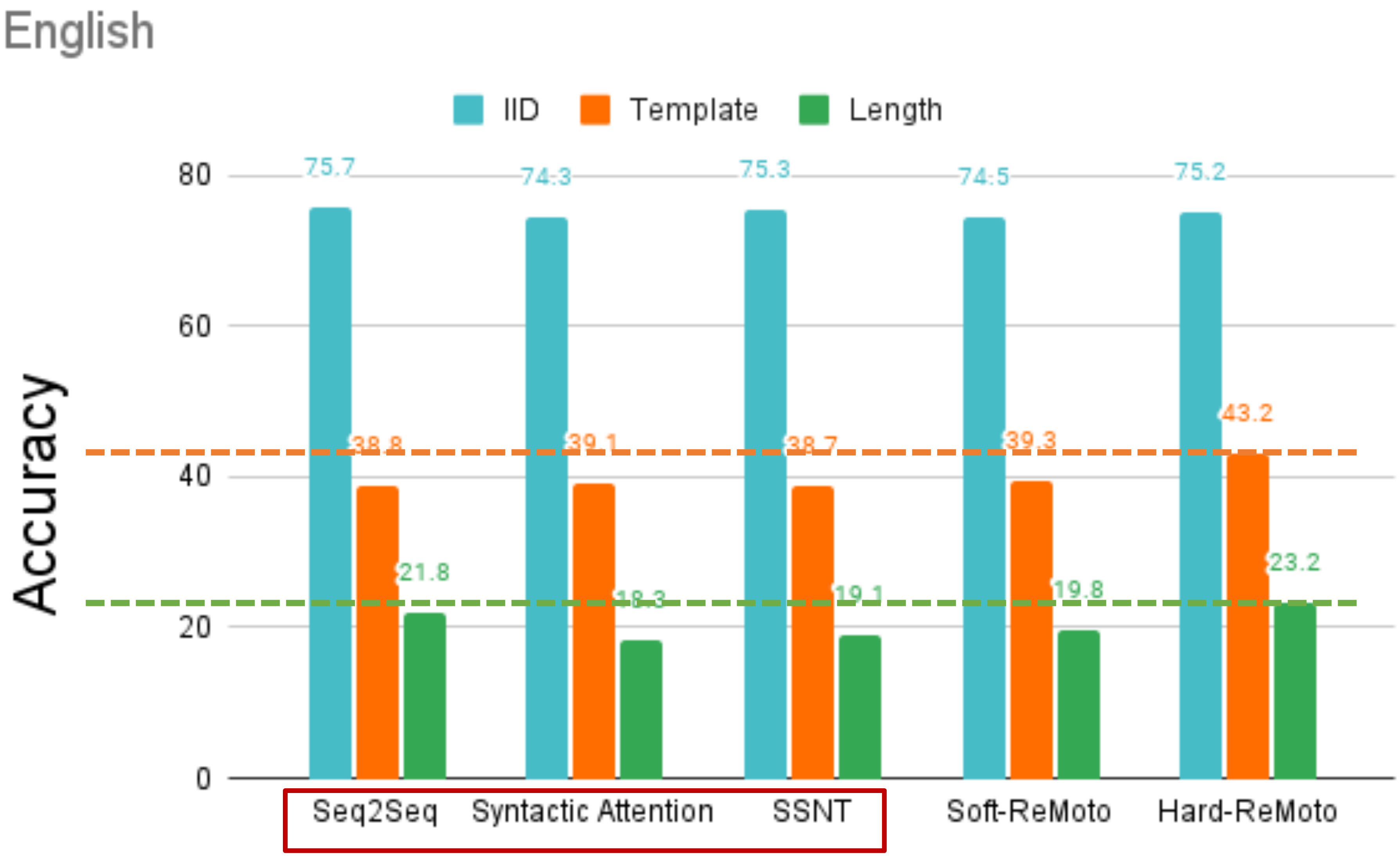
Experiment: Semantic Parsing

- Mapping natural language utterances to executable programs

Input:	how many states do not have rivers ?
Output:	count(exclude(state(all), loc_1(river(all))))

- Splits
  - IID split: a standard split
  - Template split: training and test examples have *disjoint templates*
  - Length split: test examples are *longer* than training examples

Results



Syntactic Attention: [Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. 2019]

Summary

- A seq2seq model for NLP tasks that accounts for *latent non-monotonic segment-level alignments*.
- Efficient algorithms for exact marginal and MAP inference with separable permutations, allowing for end-to-end training
- Better systematic generalization on both synthetic and real NLP tasks.
- Code and data are available at https://github.com/berlino/tensor2struct-public