Unsupervised Natural Language Parsing

Kewei Tu ShanghaiTech University
Yong Jiang Alibaba DAMO Academy
Wenjuan Han National University of Singapore
Yanpeng Zhao University of Edinburgh

Tutorial Overview

1.	Introduction	(Kewei)
2.	Generative Approaches	(Kewei, Yong)
3.	Discriminative Approaches	(Wenjuan)
4.	Special Topics	(Yanpeng)
5.	Summary	(Kewei)

Part 4: Special Topics

Outline

- Lexicalized Grammars
    - Head-driven grammar learning
- Multimodal Grammar Induction
    - Regularities in multimodal data
- Structurally Constrained Language Model
    - Structural dependencies for the next word prediction
- Syntax Probes
    - Parameter-free grammar induction
- Multilingual Grammar Induction
    - Similarities between languages

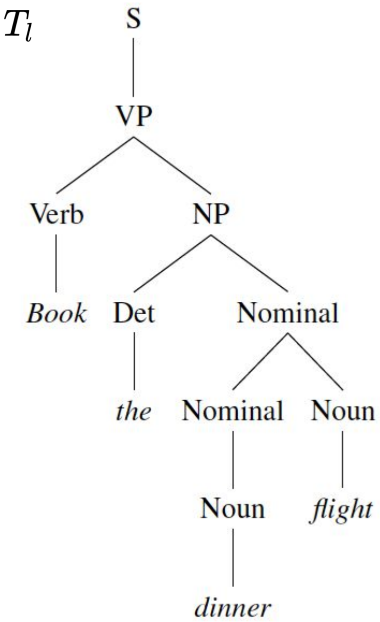
Outline

- Lexicalized Grammars
    - Head-driven grammar learning

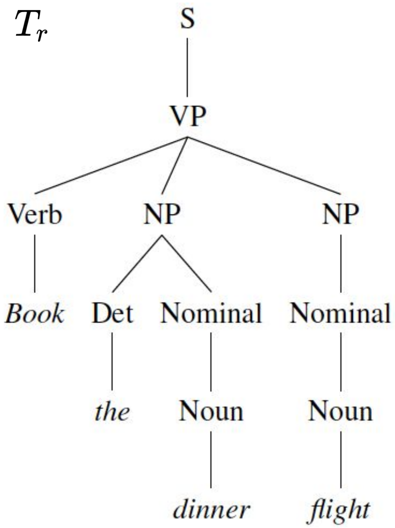
Lexicalized Grammars

Ambiguity prevails in sentences.

T_l



T_r



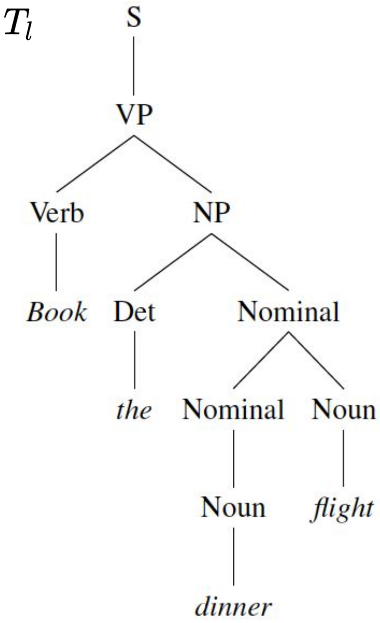
T_l : book a flight which serves dinner.

T_r : book a flight for “the dinner”.

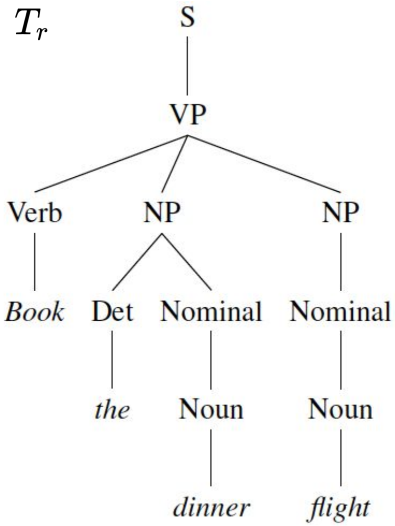
Lexicalized Grammars

PCFGs for disambiguation.

T_l



T_r

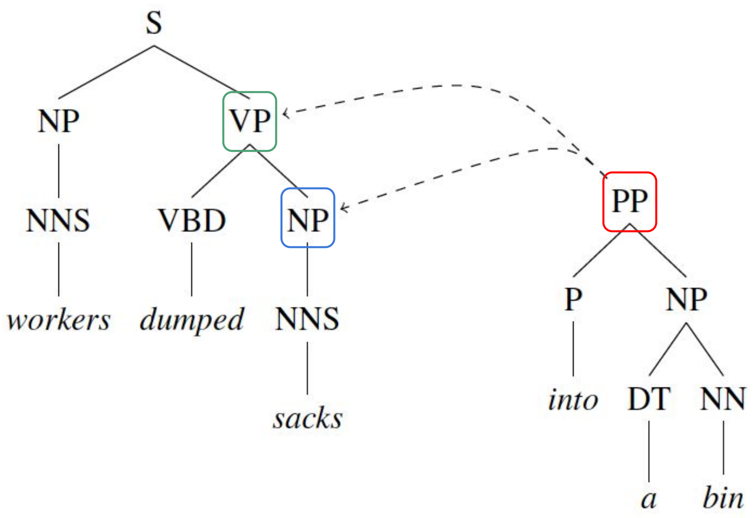


PCFGs assign each tree a probability.

Under PCFGs $p(T_l) > p(T_r)$ as the left parse is more sensible.

Lexicalized Grammars

PCFGs are inadequate to resolve ambiguity of sentences.



The prepositional phrase (PP) can be attached to either the verb phrase (VP) or the nominal phrase (NP).

The resulting trees have *very* similar probabilities.

$$\text{VP} \rightarrow \text{VBD NP PP}$$

Attached to VP

small diff.

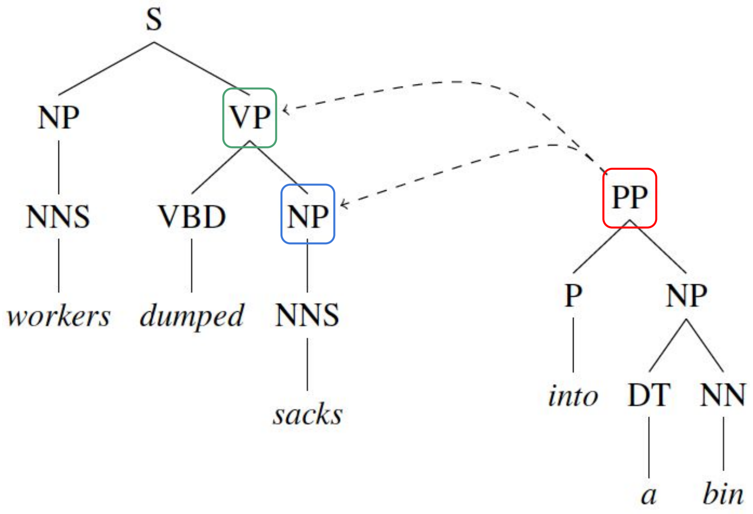
$$\text{NP} \rightarrow \text{NP PP}$$

$$\text{VP} \rightarrow \text{VBD NP}$$

Attached to NP

Lexicalized Grammars

PCFGs are inadequate to resolve ambiguity of sentences.



PP can be attached to either VP or NP.

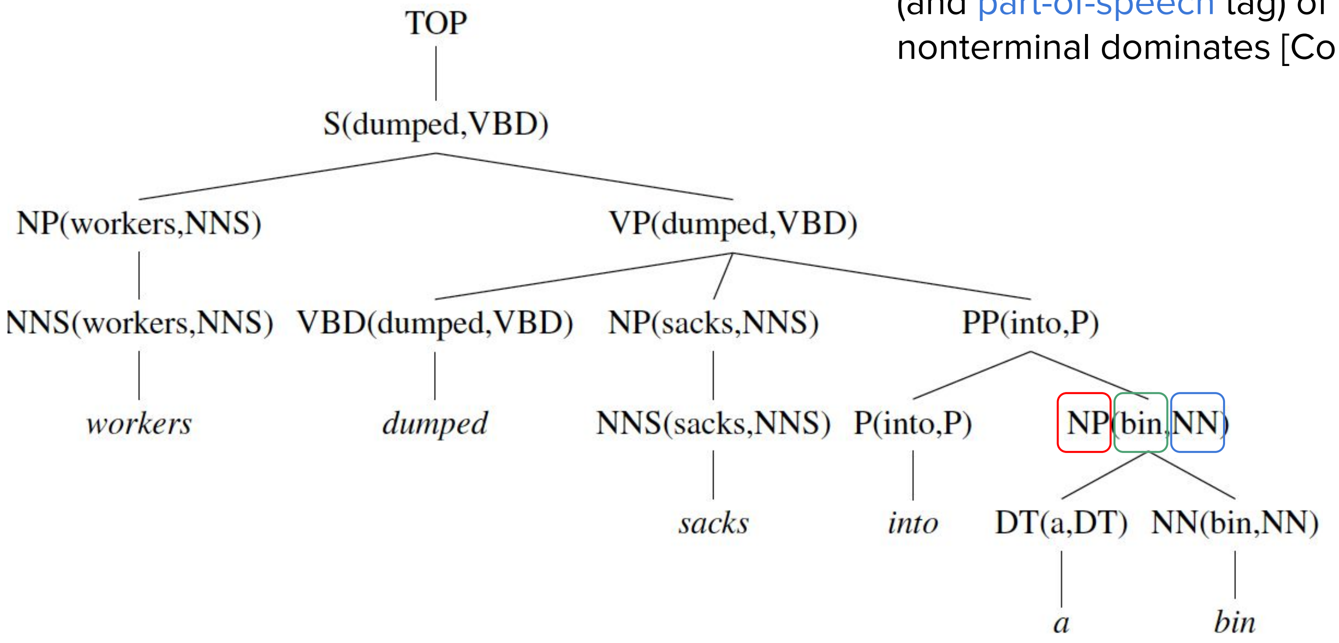
The resulting trees have *very* similar probabilities.

“*into*” is more likely to bind to “*dump*” than “*sack*”.

Lexicalized Grammars

Lexicalized PCFGs for disambiguation.

Each nonterminal is annotated by the headword (and part-of-speech tag) of the phrase which the nonterminal dominates [Collins et al., 2003].

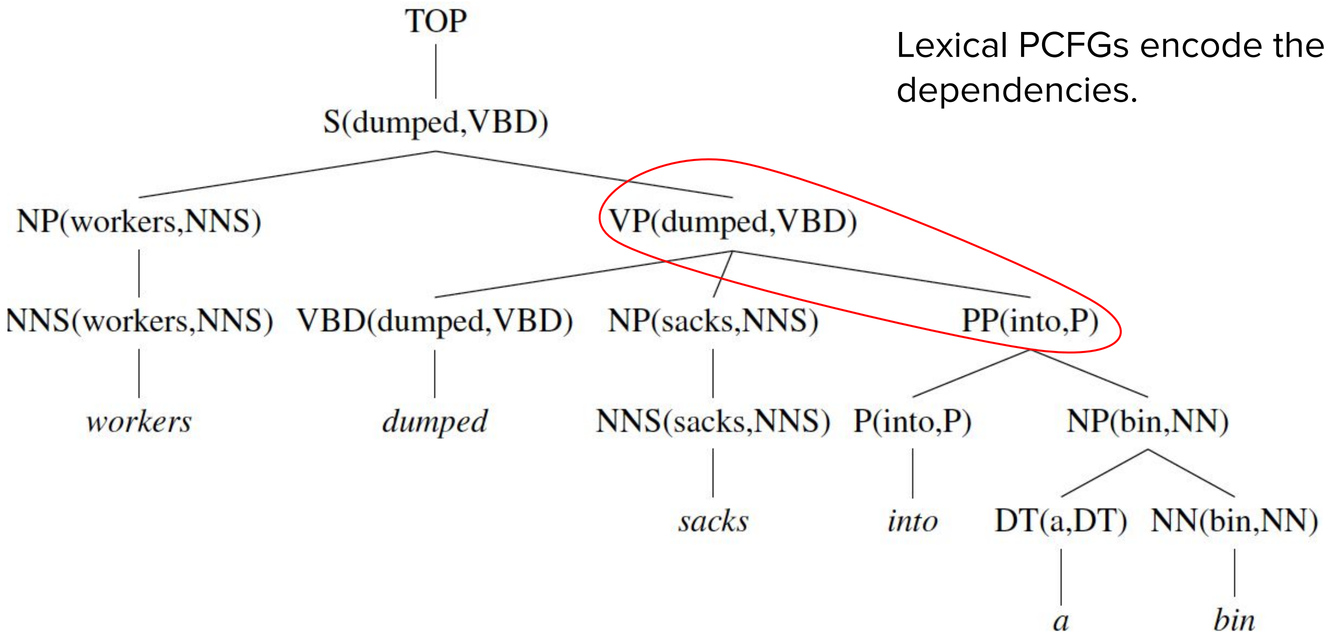


Lexicalized Grammars

Lexicalized PCFGs for disambiguation.

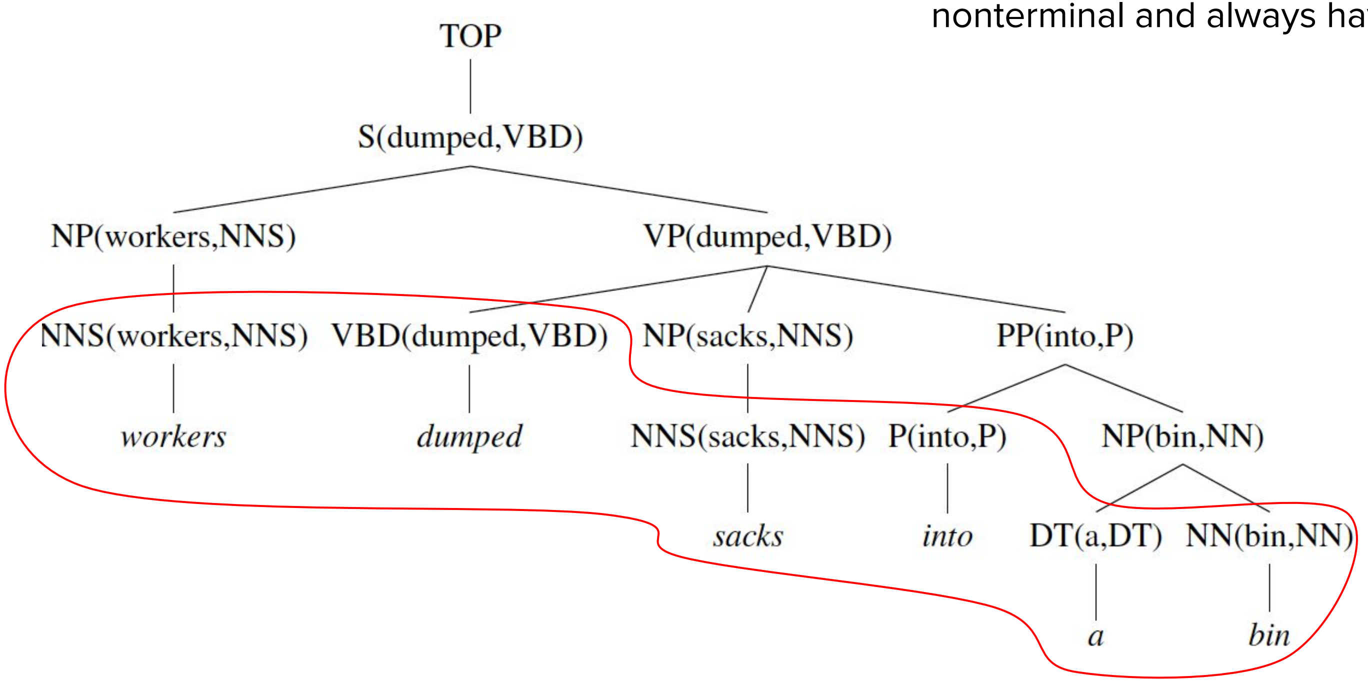
“*into*” is more likely to bind to “*dump*” than “*sack*”.

Lexical PCFGs encode the desired bi-lexical dependencies.



Lexicalized Grammars

Lexical rules in lexicalized PCFGs.

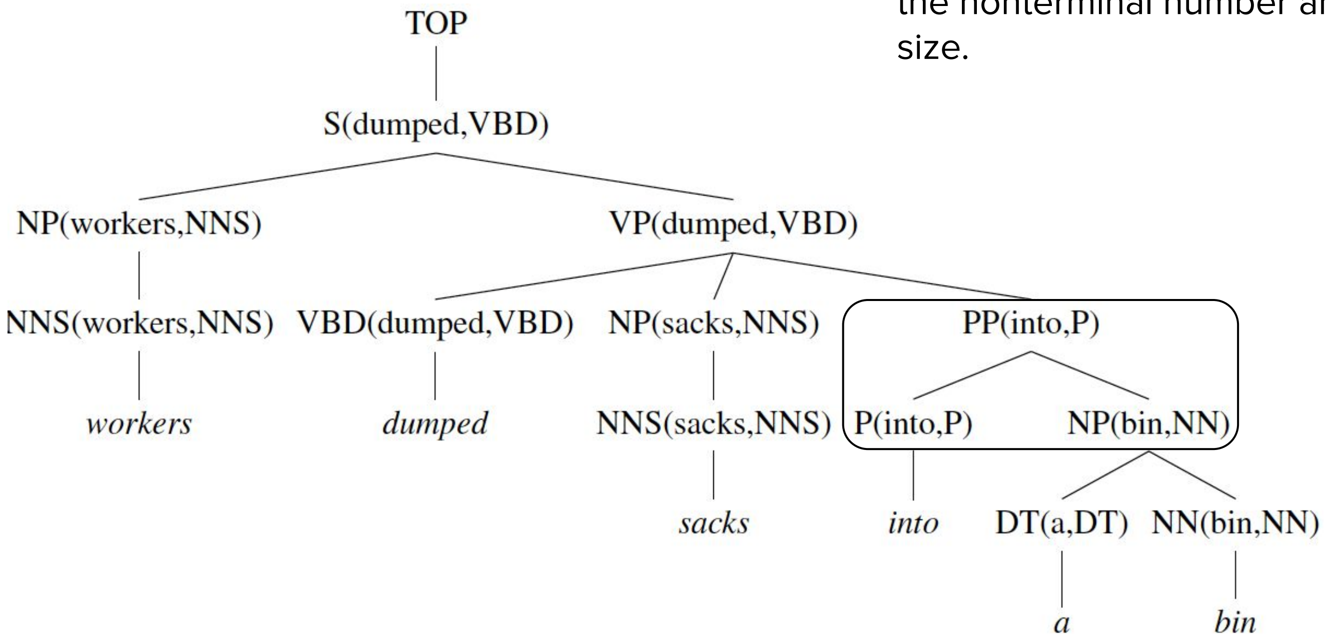


Lexical rules generate a word from an annotated nonterminal and always have the probability of 1.

Lexicalized Grammars

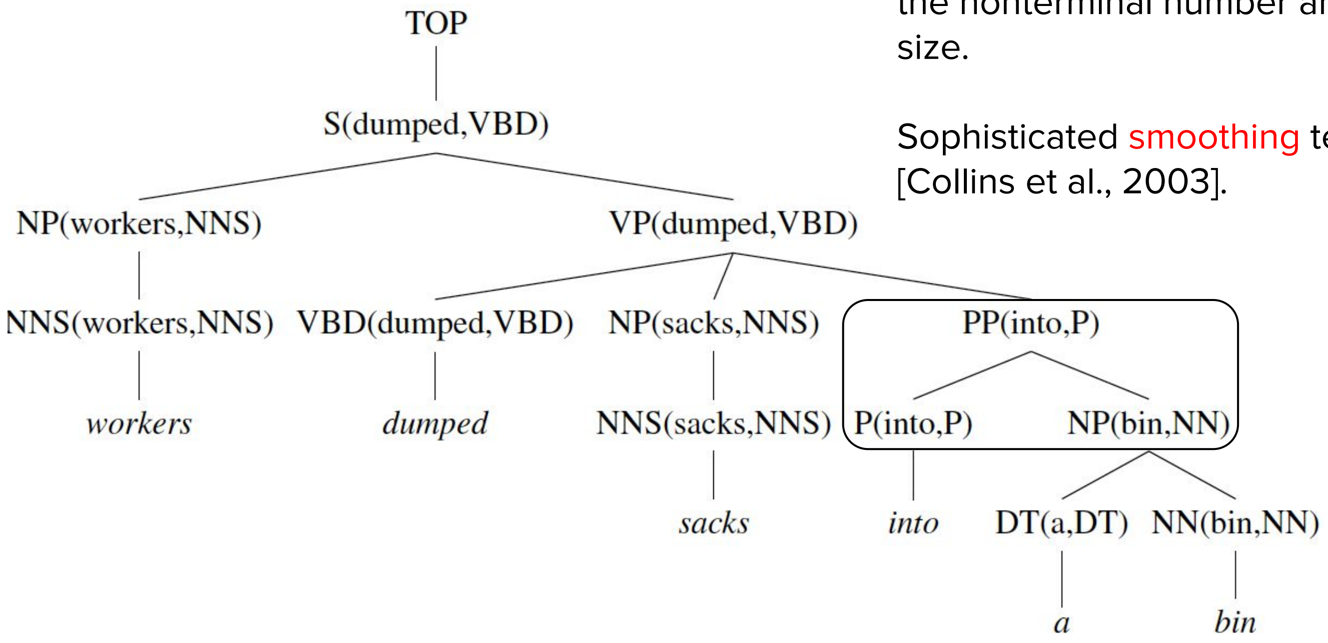
Practical issues of Lexical PCFGs.

Binary rules are too sparse: $O(n^3 v^2)$ where n is the nonterminal number and v is the vocabulary size.



Lexicalized Grammars

Practical issues of Lexical PCFGs.



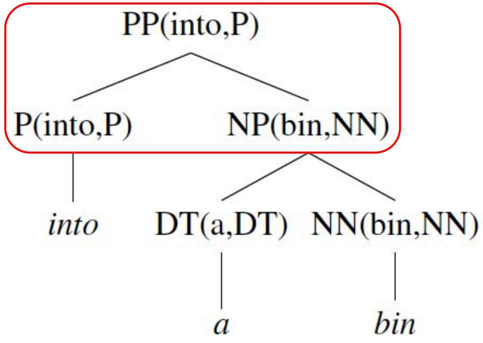
Binary rules are too sparse: $O(n^3v^2)$ where n is the nonterminal number and v is the vocabulary size.

Sophisticated smoothing techniques are needed [Collins et al., 2003].

Lexicalized Grammars

Tackle the data sparsity issue: Neural Lexical PCFGs [Zhu et al. 2020].

$p(\mathrm{PP[into]} \rightarrow \mathrm{P[into]NP[bin]})$



Rule probabilities are generated by neural networks,

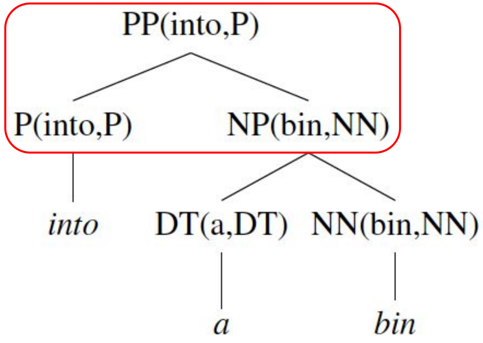
- every rule has a nonzero probability [Kim et al., 2019].

Rule r $\longrightarrow$ NN $\longrightarrow$ $p(r)$

Lexicalized Grammars

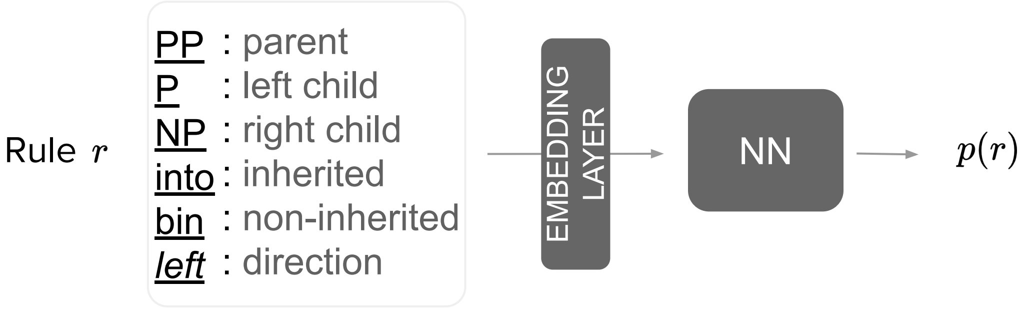
Tackle the data sparsity issue: Neural Lexical PCFGs [Zhu et al. 2020].

$p(\mathrm{PP[into]} \rightarrow \mathrm{P[into]NP[bin]})$



Nonterminals and words are represented by continuous vectors,

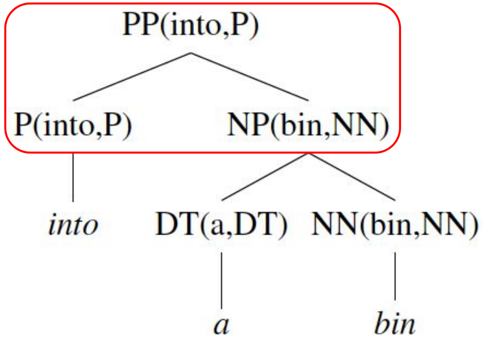
- which facilitates informed smoothing [Zhao et al., 2018].



Lexicalized Grammars

Practical issues of Lexical PCFGs.

$p(\mathrm{PP}[\mathrm{into}] \rightarrow \mathrm{P}[\mathrm{into}]\mathrm{NP}[\mathrm{bin}])$



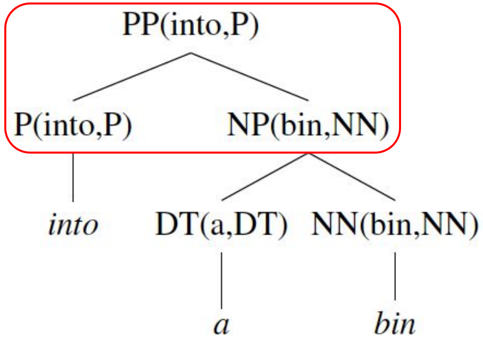
High time and space complexities:

- space complexity: $O(n^3v^2)$ where n is # of nonterminals and v is the vocabulary size.
- time complexity: $O(n^3l^5)$ where l is the sentence length (by naive application of the inside algorithm).

Lexicalized Grammars

Reduce the complexities: Neural Lexical PCFGs [Zhu et al. 2020].

$p(\mathrm{PP}[\mathrm{into}] \rightarrow \mathrm{P}[\mathrm{into}]\mathrm{NP}[\mathrm{bin}])$



Factorized rule probability:

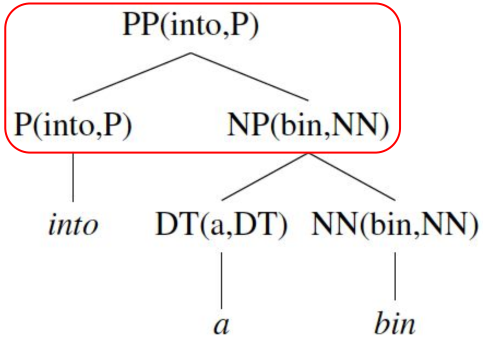
- $p(\mathrm{PP}[\mathrm{into}] \rightarrow \mathrm{P}[\mathrm{into}]\mathrm{NP}[\mathrm{bin}])$

  $= p(\mathrm{P}, \mathrm{NP}, \mathrm{dir{=}left} \mid \mathrm{PP}, \mathrm{into})p(\mathrm{bin} \mid \mathrm{NP})$
- such that the number of rules is reduced;
- and the caching trick can be used to reduce computation.

Lexicalized Grammars

Joint induction of constituency and dependency grammars [Zhu et al. 2020].

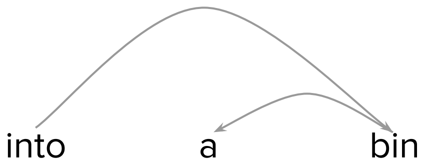
$$p(\mathrm{PP}[\mathrm{into}] \to \mathrm{P}[\mathrm{into}]\mathrm{NP}[\mathrm{bin}])$$



phrase structure

Lexicalized PCFGs encode lexical dependencies [Collins 2003].

- dependency grammars can be induced as a byproduct.



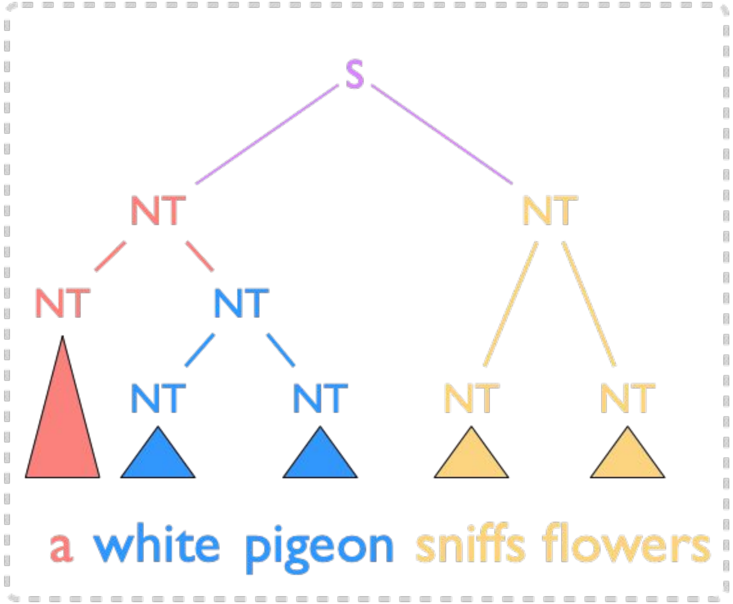
dependency structure

Outline

- Multimodal Grammar Induction
  - Regularities in multimodal data

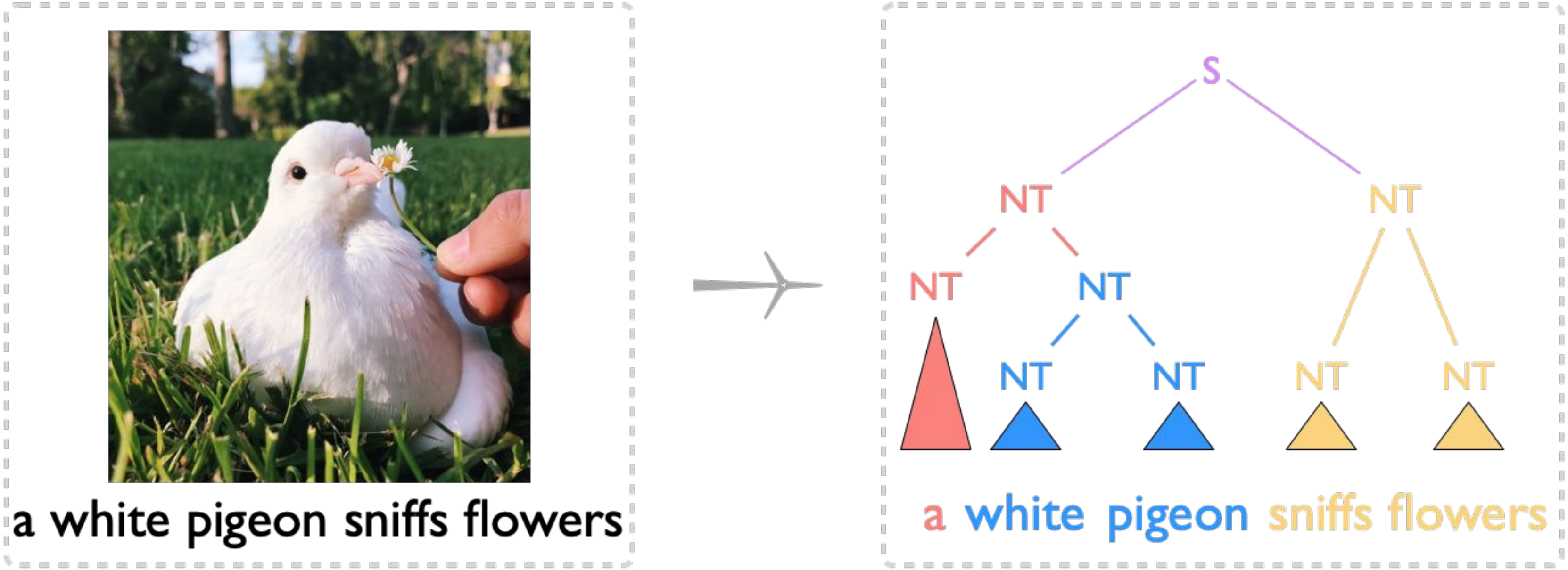
Visually Grounded Grammar Induction

Can visual groundings help us induce syntactic structure?



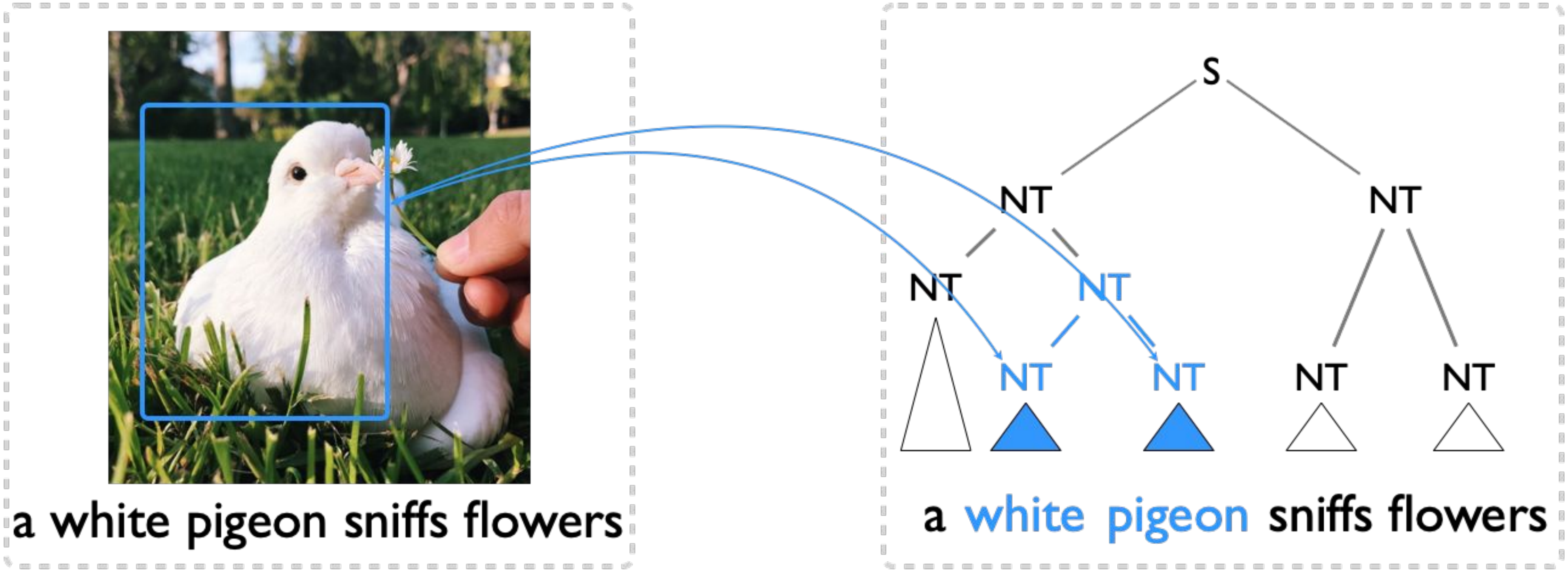
Visually Grounded Grammar Induction

The task: inducing phrase-structure grammars from sentences and their visual groundings.



Multimodal Grammar Induction

Exploiting regularities between the semantic content of the image and the syntactic structure.



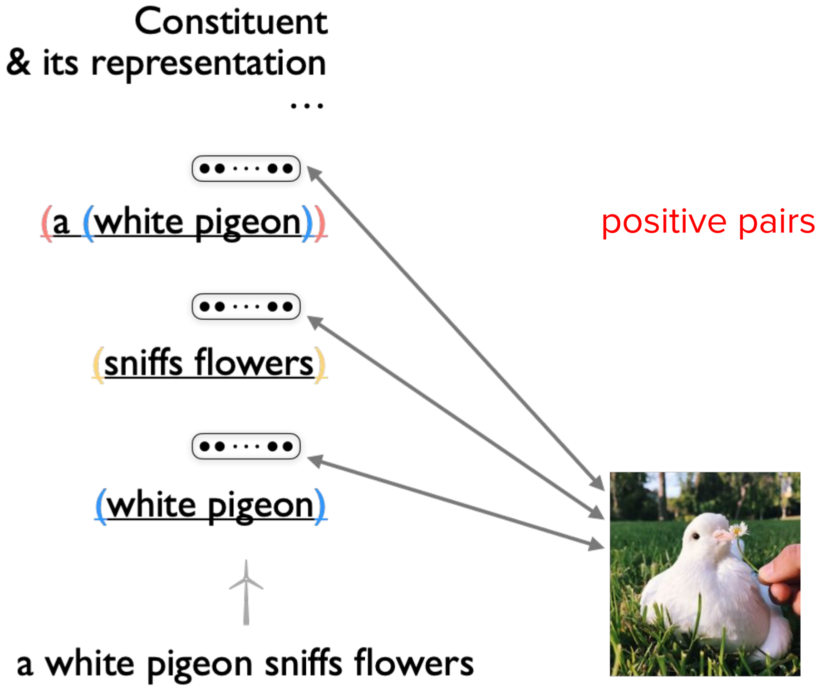
Multimodal Grammar Induction

Visually grounded neural syntax learner (VG-NSL) [Shi et al., 2019].

- Visual and textual representation model (capturing the regularities)
- Parsing model (inducing tree structures)

Multimodal Grammar Induction: VG-NSL

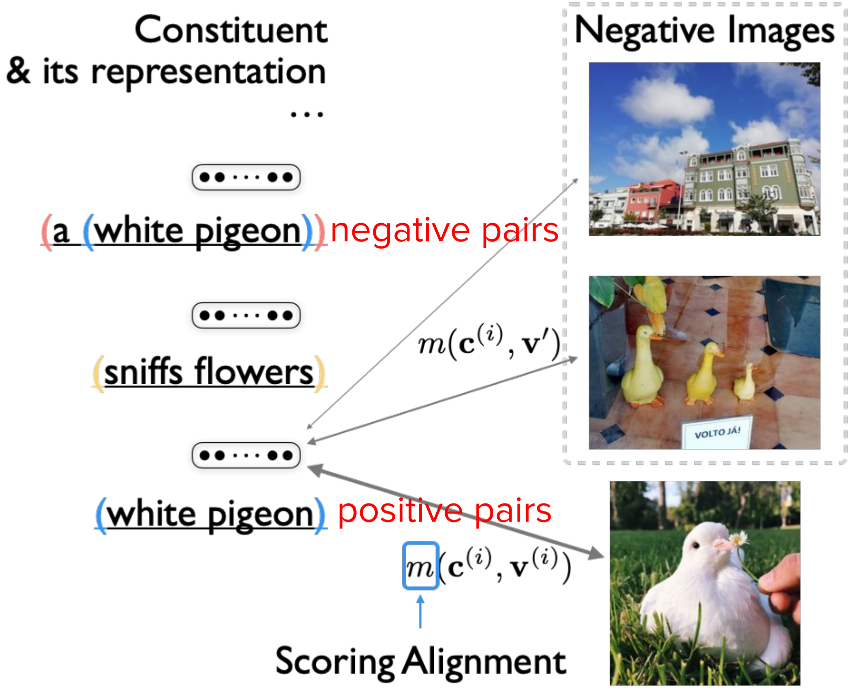
Optimize visual and textual representations to capture the regularities.



Optimized via contrastive learning...

Multimodal Grammar Induction: VG-NSL

Optimize visual and textual representations to capture the regularities.

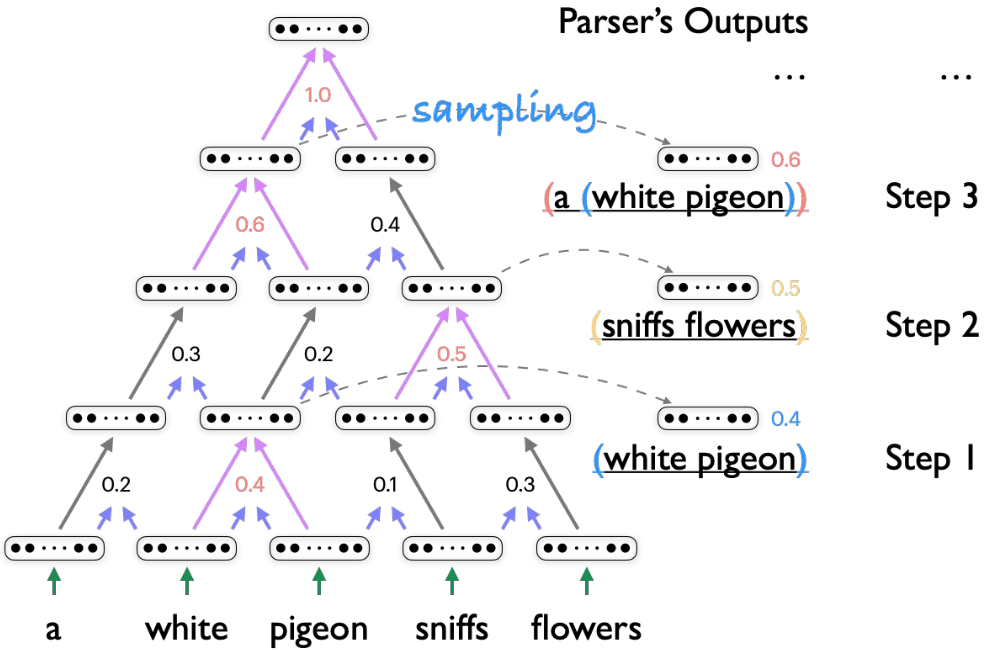


Optimized via contrastive learning...

... such that positive pairs score higher than negative pairs.

Multimodal Grammar Induction: VG-NSL

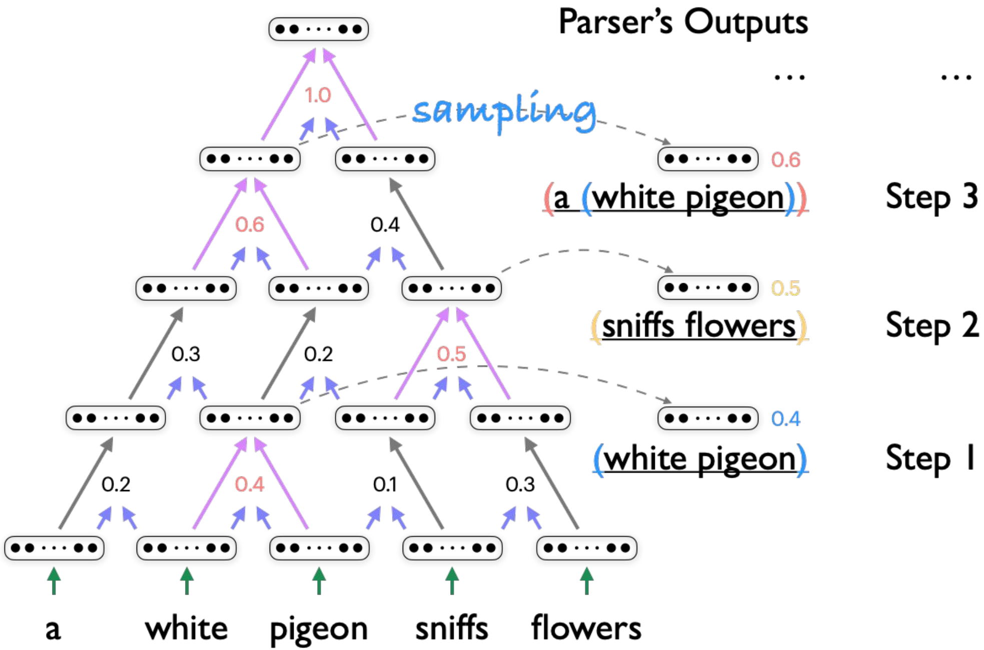
Optimize the parsing model to produce plausible trees.



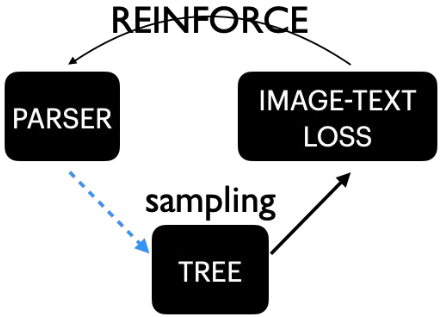
An easy-first greedy parser [Goldberg et al., 2010]...

Multimodal Grammar Induction: VG-NSL

Optimize the parsing model to produce plausible trees.



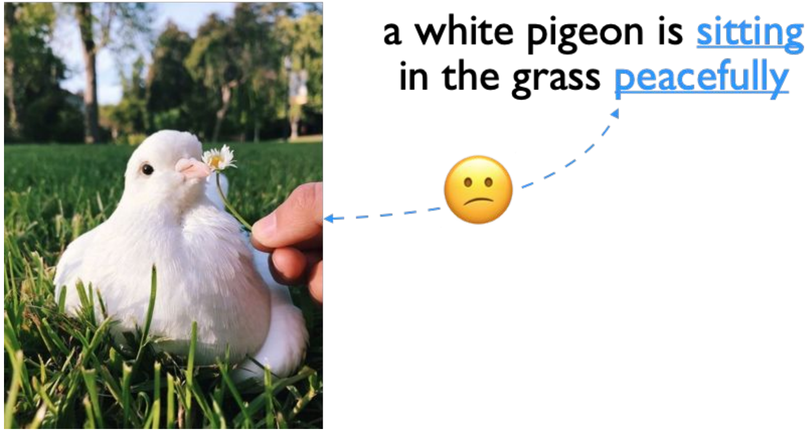
An easy-first greedy parser [Goldberg et al., 2010] is optimized via REINFORCE.

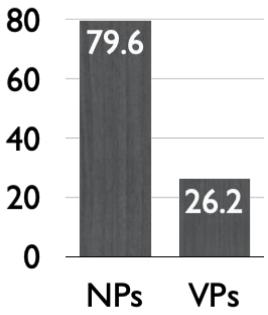


Multimodal Grammar Induction: VG-NSL

Practical issues with VG-NSL.

- REINFORCE suffers from large variance in gradient estimation
- No obvious visual signals for certain syntactic phenomena [Shi et al., 2019; Kojima et al., 2020]

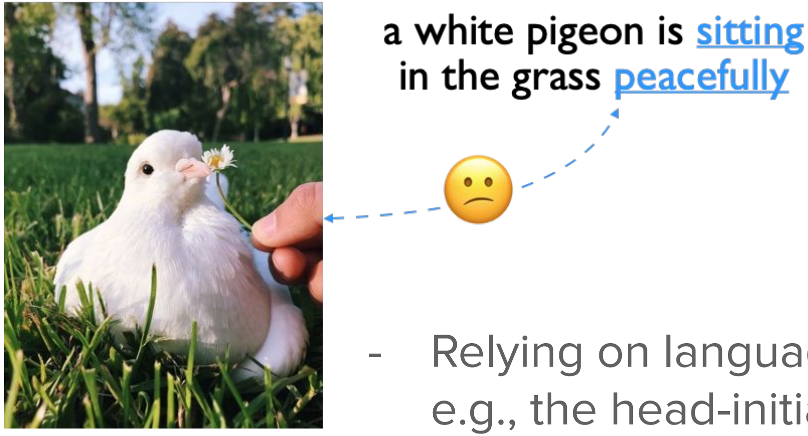


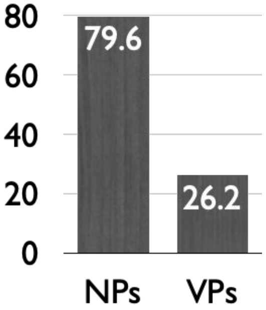


Multimodal Grammar Induction: VG-NSL

Practical issues with VG-NSL.

- REINFORCE suffers from large variance in gradient estimation
- No obvious visual signals for certain syntactic phenomena [Shi et al., 2019; Kojima et al., 2020]



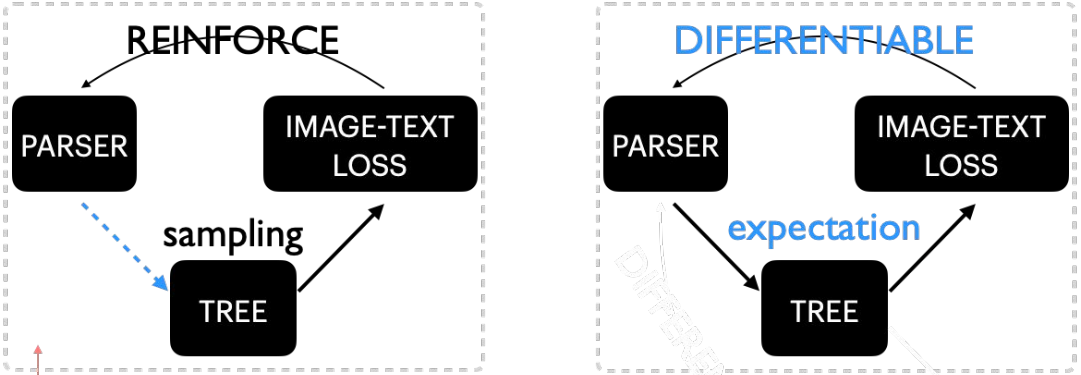


- Relying on language-specific priors to alleviate the issues, e.g., the head-initial preference in English [Baker, 2008]

Multimodal Grammar Induction

Visually grounded compound PCFGs (VC-PCFG) [Zhao et al., 2020].

~~Noisy gradient estimation~~

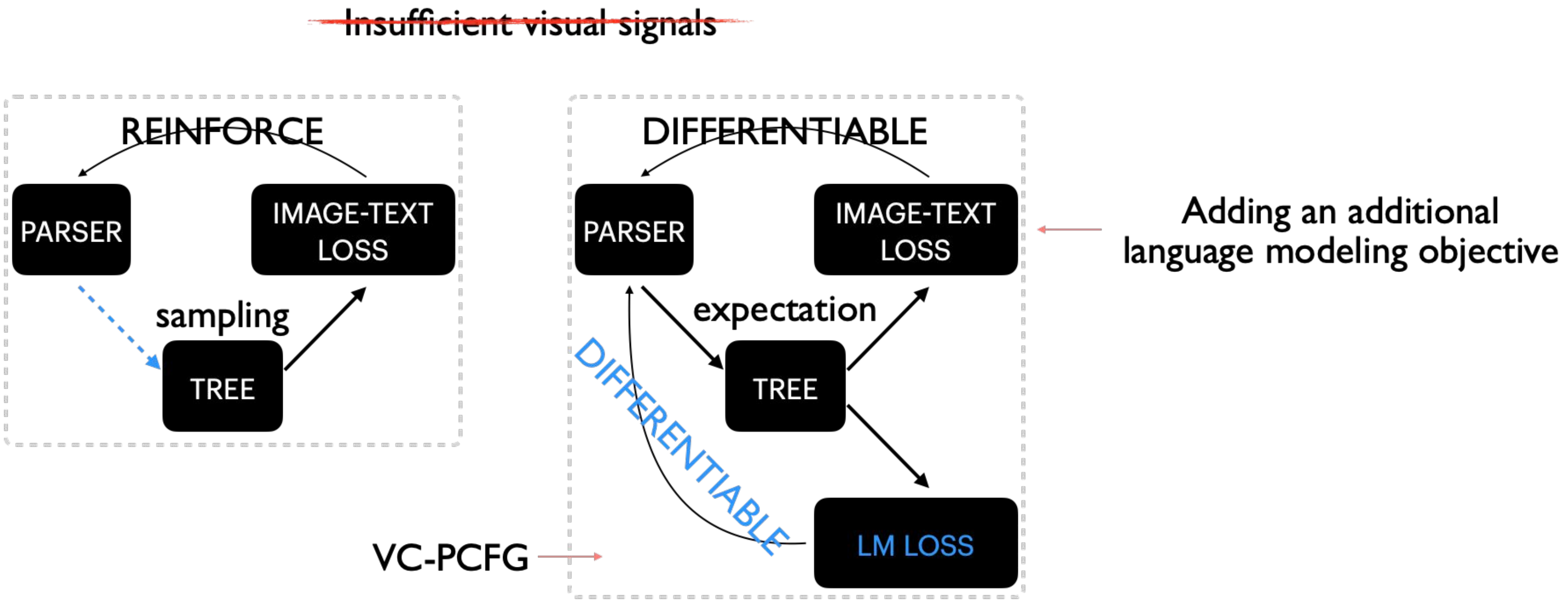


Easy-first parser replaced by (Compound) PCFGs

VG-NSL

Multimodal Grammar Induction

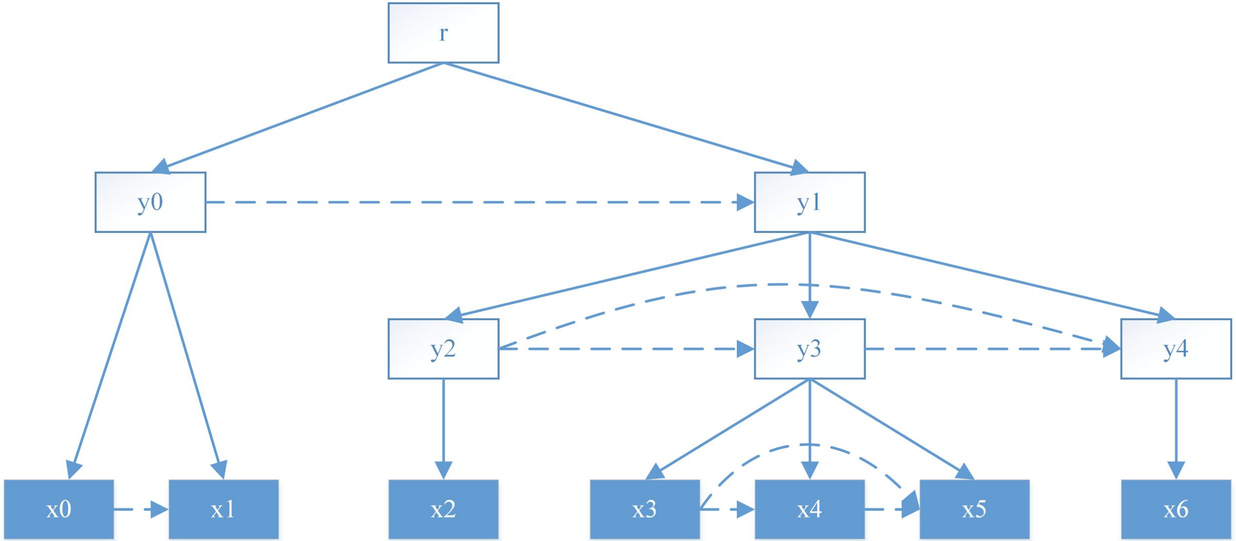
Visually grounded compound PCFGs (VC-PCFG) [Zhao et al., 2020].



Outline

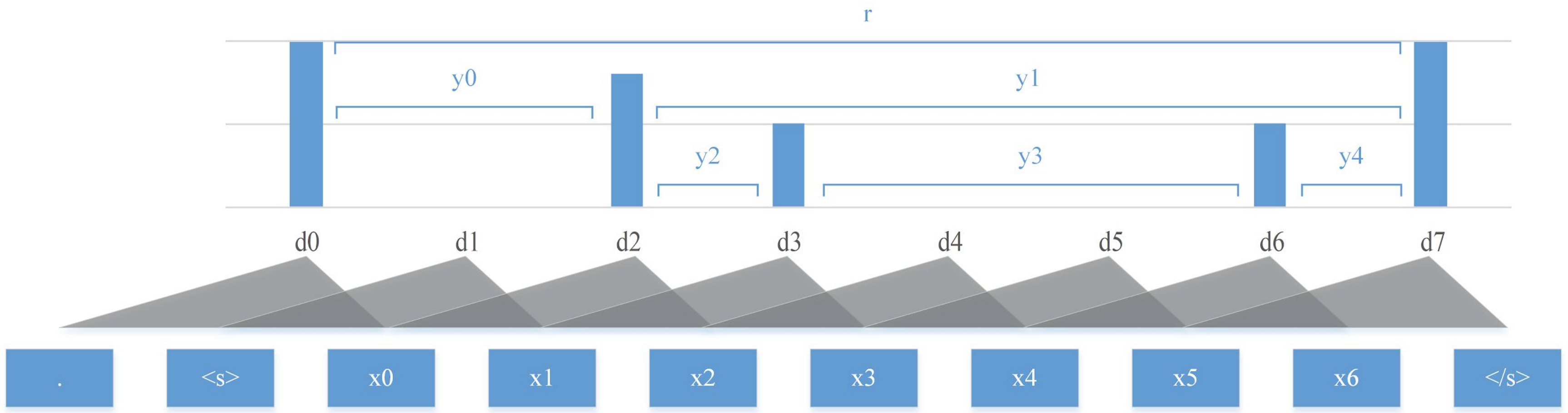
- Structurally Constrained Language Model
  - Structural dependencies for the next word prediction

Structurally Constrained Language Models



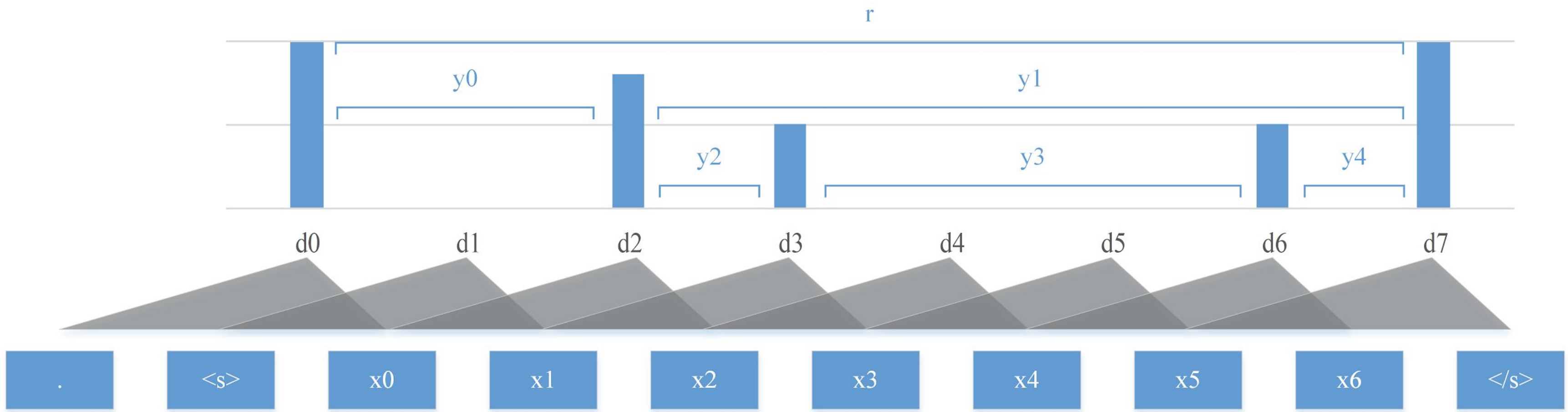
Parsing with Syntactic Distance

Syntactic distance measures “surprisal” between adjacent words [Shen et al., 2017].



Parsing with Syntactic Distance

A larger syntactic distance indicates that there is more likely to be a constituent boundary between two adjacent words.



Parsing with Syntactic Distance

Top-down decoding via recursive binary splitting [Dyer et al., 2019].

Input sentence $w = w_1 w_2 \ldots w_n$ and syntactic distance $d = d_1 d_2 \ldots d_{n-1}$.

$\text{def parse}(w, d)$:

$\quad \text{if } d = []$:

$\quad\quad \text{node} = \text{LEAF}(w_1)$

$\quad \text{else}$:

$\quad\quad k = \arg\max_i d_i$ ← splitting point

$\quad\quad r_{\text{child}} = \text{parse}(w_{1:k}, d_{1:k-1})$ ← left subtree

$\quad\quad l_{\text{child}} = \text{parse}(w_{k+1:\text{end}}, d_{k+1:\text{end}})$ ← right subtree

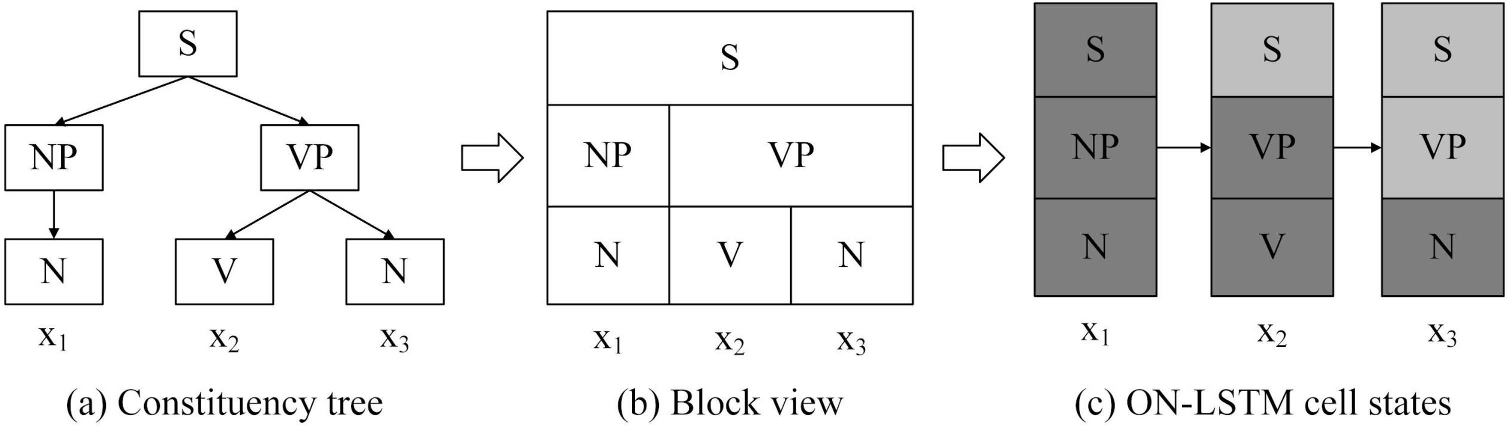
$\quad\quad \text{node} = \text{NODE}(l_{\text{child}}, r_{\text{child}})$

$\quad \text{return node}$

Structurally Constrained Language Models (ON-LSTM)

Ordered neurons LSTM (ON-LSTM) [Shen et al., 2018].

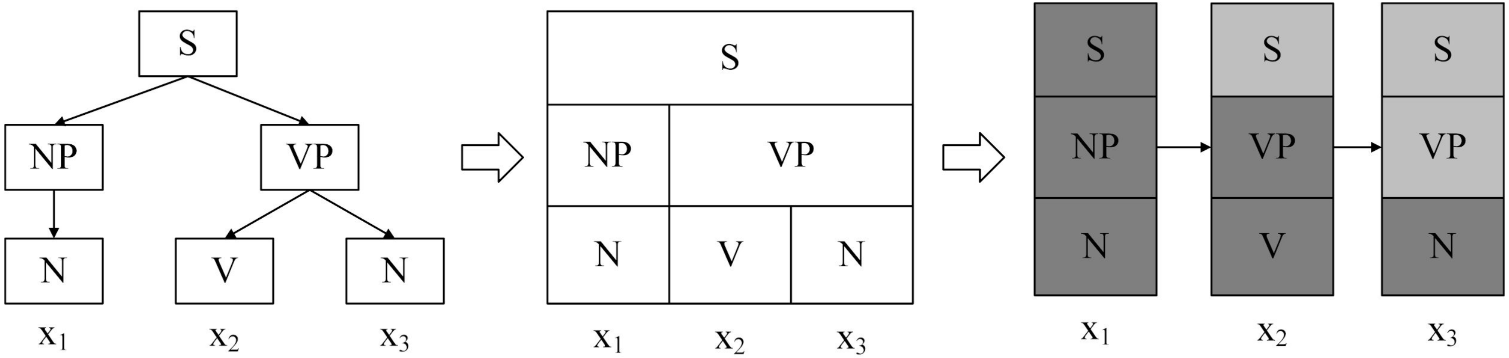
- Integrate tree structures into recurrent neural networks (e.g., LSTMs).



Structurally Constrained Language Models (ON-LSTM)

Ordered neurons LSTM (ON-LSTM) [Shen et al., 2018].

- Neurons are ordered from bottom to up indicating different updating rates.
- Control information flow via ordered forget and input gate neurons.

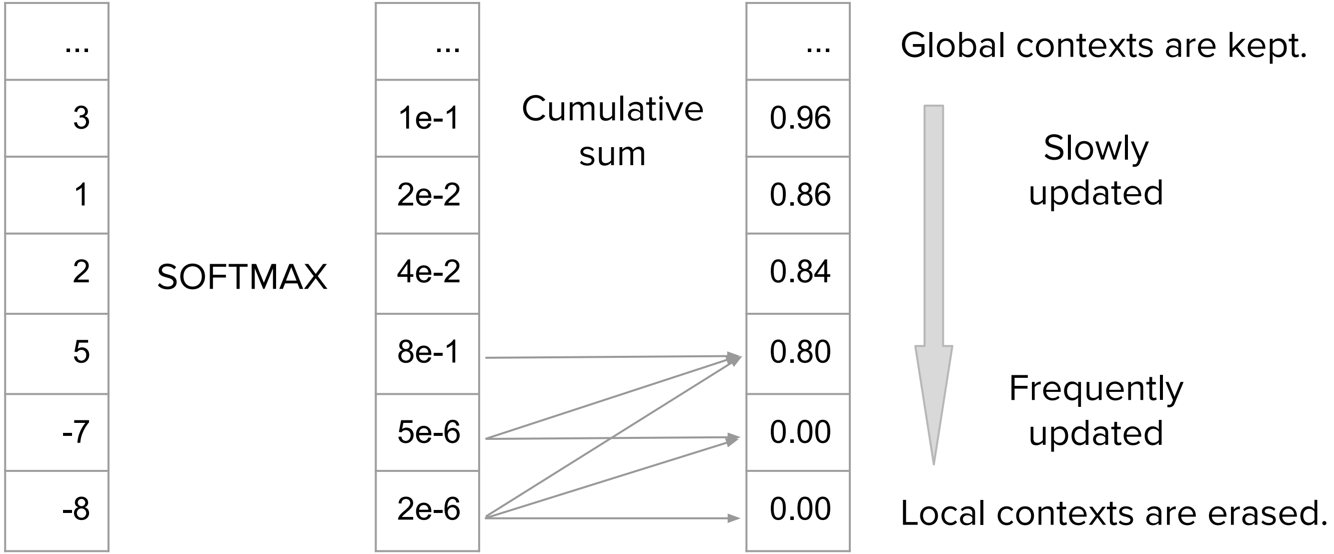


(a) Constituency tree (b) Block view (c) ON-LSTM cell states

Structurally Constrained Language Models (ON-LSTM)

Control information flow via forget gate neurons.

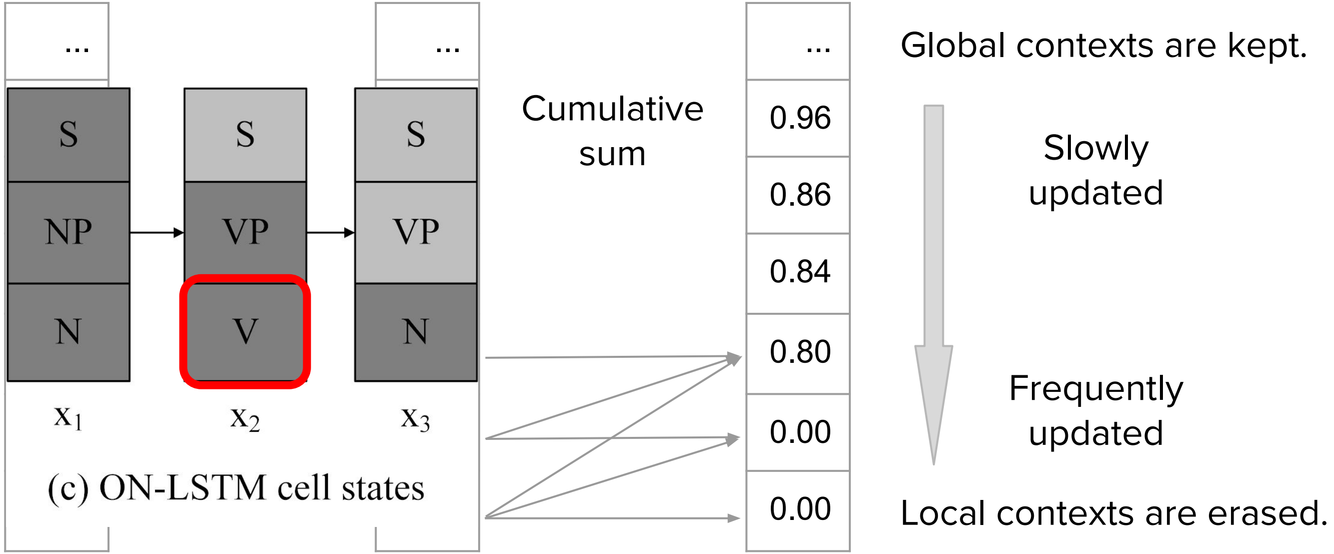
- Neurons are ordered from bottom to up indicating different updating rates.



Structurally Constrained Language Models (ON-LSTM)

Control information flow via forget gate neurons.

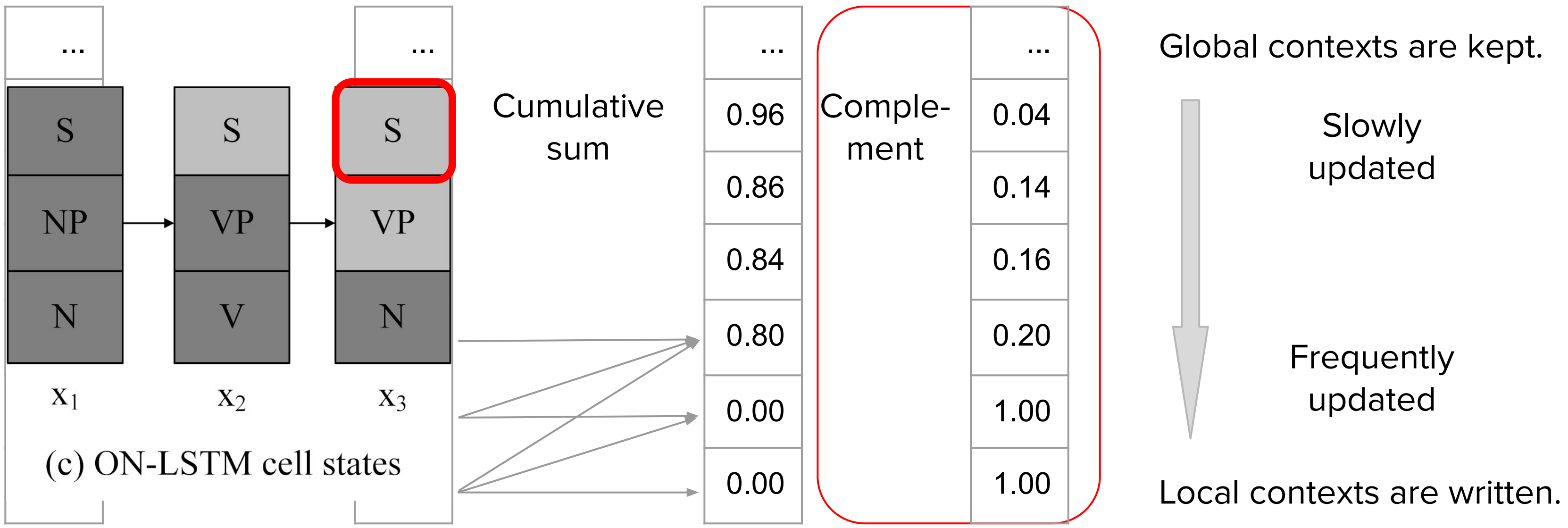
- Neurons are ordered from bottom to up indicating different updating rates.



Structurally Constrained Language Models (ON-LSTM)

Control information flow via input gate neurons.

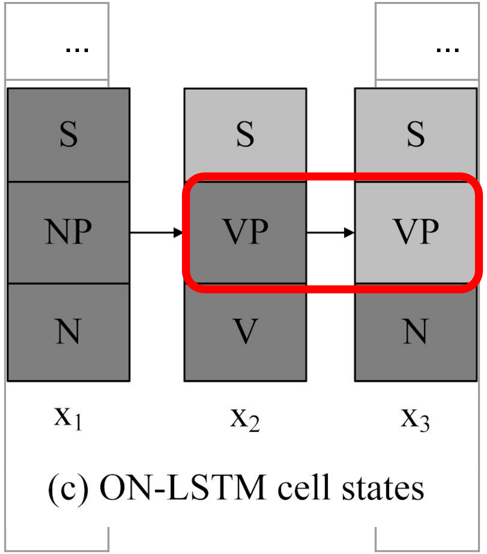
- Neurons are ordered from bottom to up indicating different updating rates.



Structurally Constrained Language Models (ON-LSTM)

Control information flow via forget (F) and input (I) gate neurons.

- The multiplication of F and I gate neurons encodes incomplete constituents.

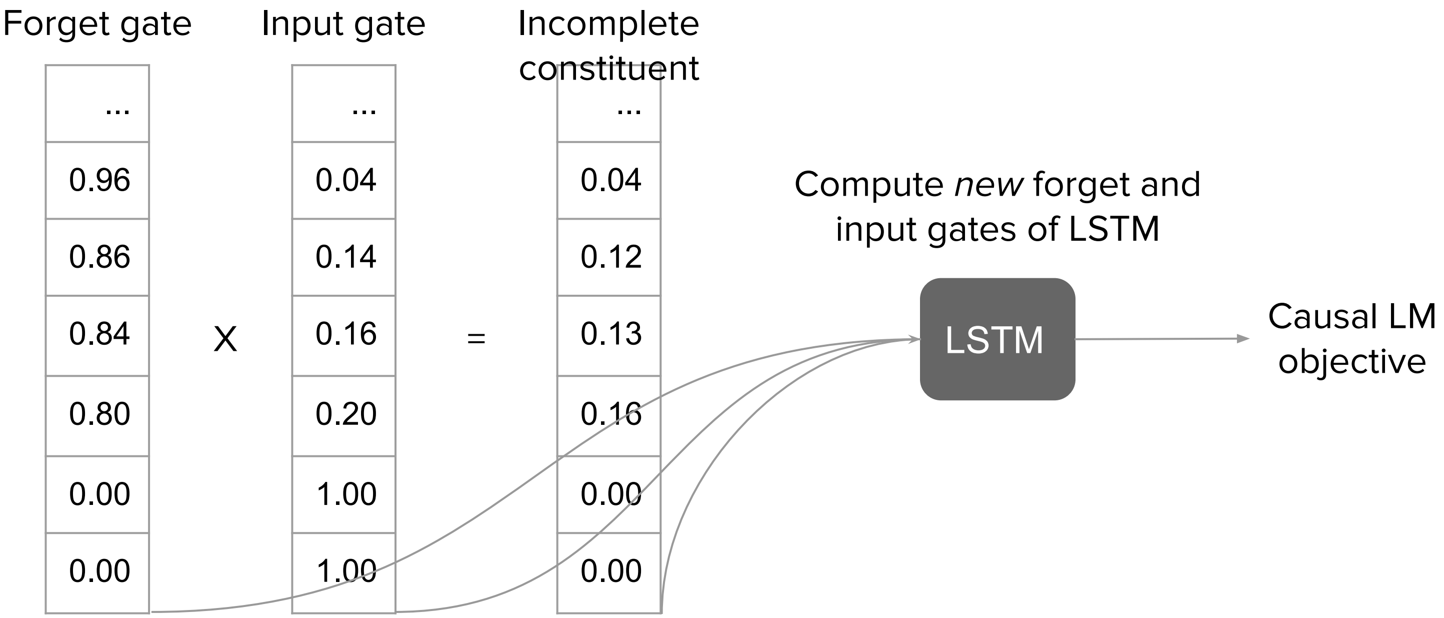


(c) ON-LSTM cell states

Forget gate		Input gate			
...		...		...	
0.96		0.04		0.04	
0.86		0.14		0.12	
0.84	X	0.16	=	0.13	VP
0.80		0.20		0.16	
0.00		1.00		0.00	
0.00		1.00		0.00	

Structurally Constrained Language Models (ON-LSTM)

ON-LSTM is trained as a language model.



Estimate Syntactic Distances (ON-LSTM)

Forget gate neurons are used to compute syntactic distances.

- The expected position which splits neurons into two halves.

		Prob. (p)	Index (k)
...		...	...
3		1e-1	6
1		2e-2	5
2	SOFTMAX	4e-2	4
5		8e-1	3
-7		5e-6	2
-8		2e-6	1

Global contexts are kept.

Slowly updated

Frequently updated

Local contexts are erased.

position

$$d = \sum_k \boxed{k} \cdot \boxed{p_k}$$

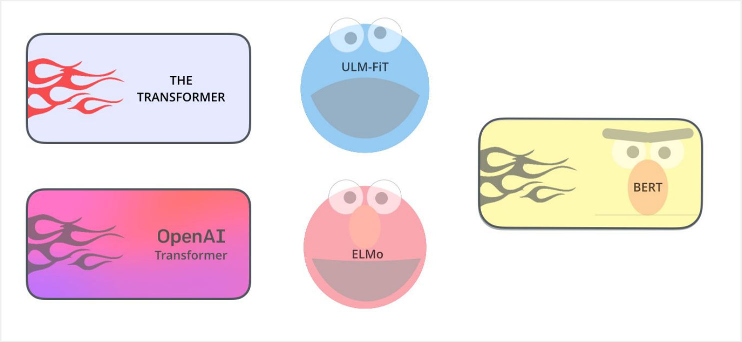
probability

Outline

- Syntax Probes
    - Parameter-free grammar induction

Syntax Probes

Pretrained models learn syntax [Shi et al., 2016, Tenney et al., 2019, Hewitt et al., 2019].

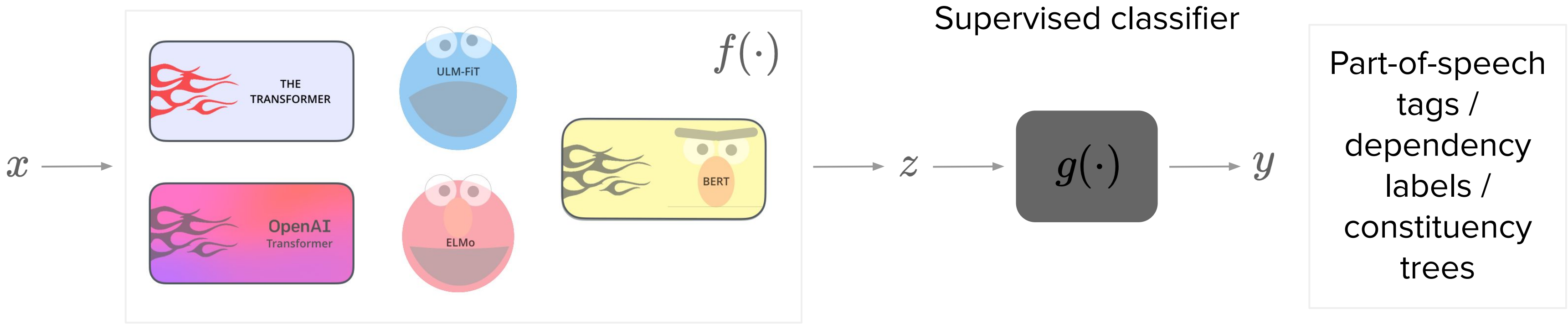


Part-of-speech tags / dependency labels / constituency trees

Images credit to http://jalammar.github.io/illustrated-bert/

Syntax Probes

How to extract syntax from pretrained models [Shi et al., 2016, Tenney et al., 2019]?



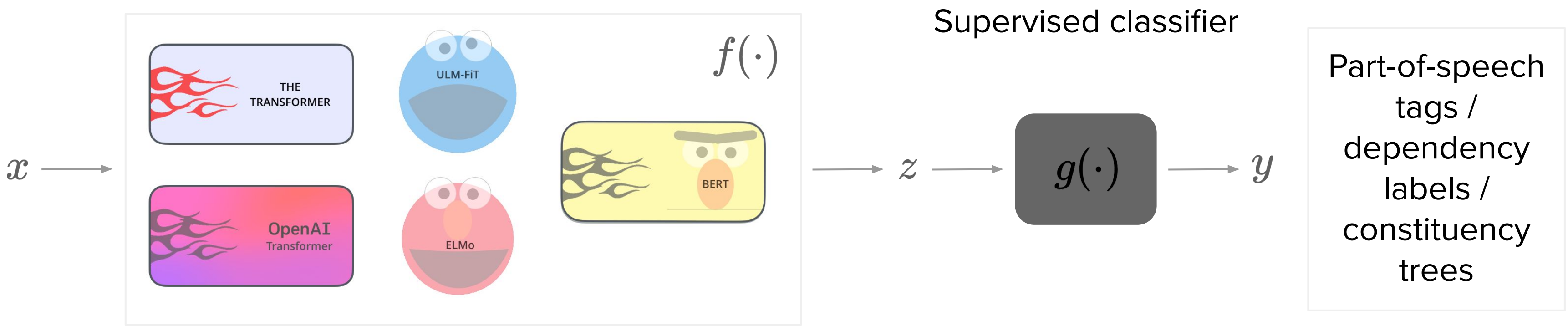
Machine translation systems /
pretrained language models.

Images credit to http://jalammar.github.io/illustrated-bert/

Syntax Probes

The classifier maximizes the mutual information between z and y.

- but does $f(\cdot)$ encode syntax or $g(\cdot)$ learn syntax when $g(\cdot)$ has a high accuracy?



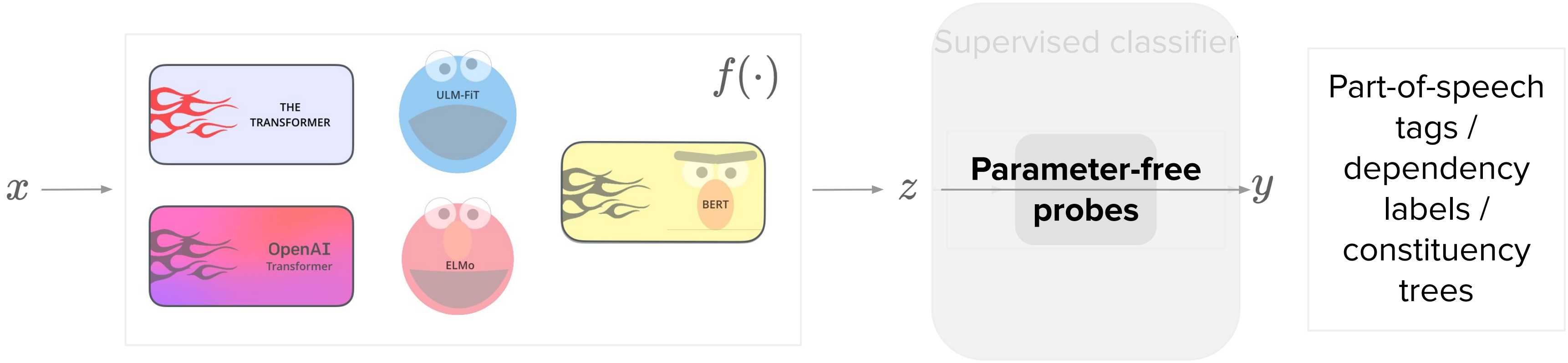
Machine translation systems /
pretrained language models.

Images credit to http://jalammar.github.io/illustrated-bert/

Syntax Probes

It is hard to explain whether $f(\cdot)$ encodes syntax or $g(\cdot)$ learns syntax.

- Parameter-free probes remove the learnable classifier.

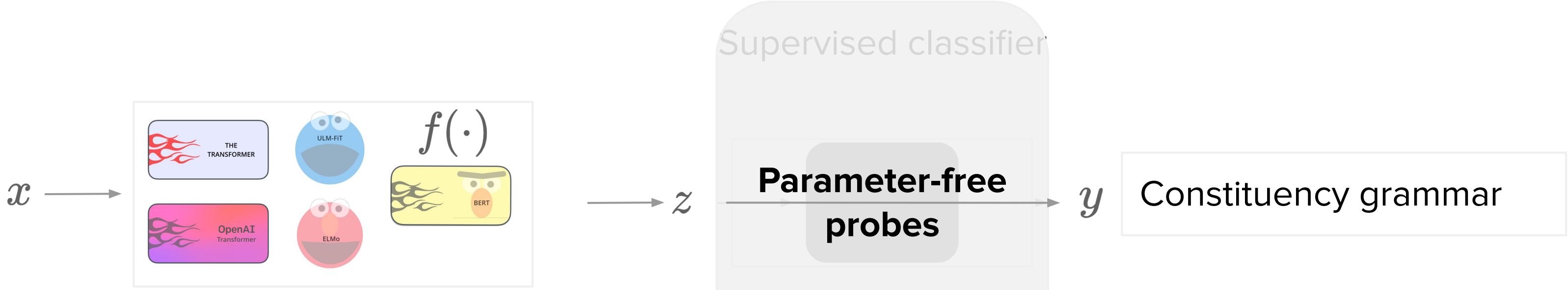


Machine translation systems / pretrained language models.

Images credit to http://jalammar.github.io/illustrated-bert/

Syntax Probes: parameter-free probes

Parameter-free probes infer syntax from pretrained models without learning.

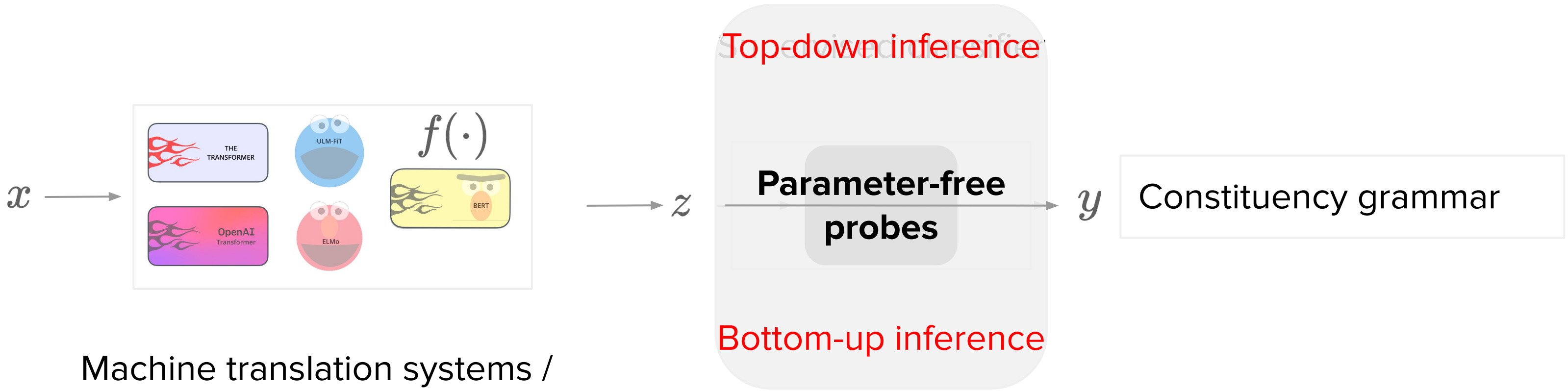


Machine translation systems / pretrained language models.

Images credit to http://jalammar.github.io/illustrated-bert/

Syntax Probes: parameter-free probes

Parameter-free probes infer syntax from pretrained models without learning.



Machine translation systems / pretrained language models.

Images credit to http://jalammar.github.io/illustrated-bert/

Syntax Probes: top-down inference

Top-down inference based on the “surprisal” (syntactic distance) between adjacent words [kim et al., 2020].

Define syntactic distance $d_i = f(g(w_i), g(w_{i+1}))$

- $f(\cdot)$ measures cosine, L1, or L2 distances when $g(\cdot)$ produces vector representations of words,

- $f(\cdot)$ measures Jensen-Shannon, or Hellinger distances when $g(\cdot)$ outputs attention distributions.

Syntax Probes: top-down inference

Top-down inference based on the "surprisal" (syntactic distance) between adjacent words [Wu et al., 2020].

- Compute an impact matrix where $f(w_i, w_j)$ encodes the impact of w_j on w_i.
    - $f(\cdot)$ measures prediction / representation difference at the $i\text{-th}$ position between masking out w_i and masking out w_i & w_j.

Syntax Probes: top-down inference

Top-down inference based on the “surprisal” (syntactic distance) between adjacent words [Wu et al., 2020].

- Compute an impact matrix where $f(w_i, w_j)$ encodes the impact of w_j on w_i.
    - $f(\cdot)$ measures prediction / representation difference at the $i\text{-th}$ position between masking out w_i and masking out w_i & w_j.
- Estimate syntactic distances using the impact matrix.
    - The average impact between words is large in the same constituent and is small in different constituents.

Syntax Probes: bottom-up inference

Bottom-up (CYK) decoding for constituency grammar induction [Li et al., 2020].

- Multi-head attentions encode interdependencies between words.
- Compute similarity for every pair of words [Kim et al., 2020]
    - $d_i = f(g(w_i), g(w_j))$ where $g(\cdot)$ is word representation; $f(\cdot)$ is distance function

Syntax Probes: bottom-up inference

Bottom-up (CYK) decoding for constituency grammar induction [Li et al., 2020].

- Multi-head attentions encode interdependencies between words.
- Compute similarity for every pair of words [Kim et al., 2020]
  - $d_i = f(g(w_i), g(w_j))$ where $g(\cdot)$ is word representation; $f(\cdot)$ is distance function
- Score every span using the similarities (similarly to Wu et al., 2020)
  - The average similarity between words is large in the same constituent and is small in different constituents.
- Parsing with the CYK algorithm [Kim et al., 2019, Cao et al., 2020]

Outline

- Multilingual Grammar Induction
  - Similarities between languages

Multilingual grammar induction

Exploit language similarities to induce grammars of different languages.

- cross-lingual word / POS representations

Multilingual grammar induction

Exploit language similarities to induce grammars of different languages.

- Independent models for different languages [Iwata et al., 2010; Jiang et al., 2019]
- Unified model for different languages [Han et al., 2019]

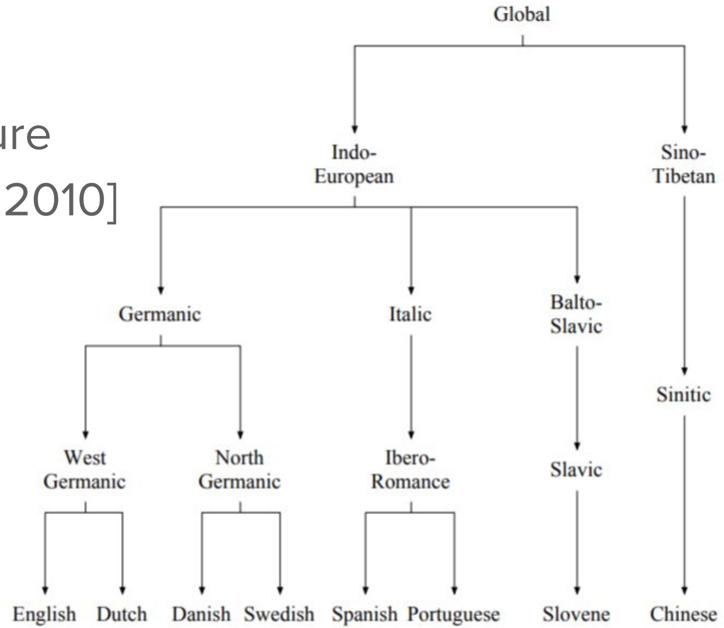
Multilingual grammar induction

Independent models for different languages [Iwata et al., 2010].

- Use *regularization* terms to encourage independent models to behave similarly [Jiang et al., 2019]

- Use *hand-crafted* phylogenetic tree to capture language similarities [Berg-Kirkpatrick et al., 2010]

phylogenetic tree →

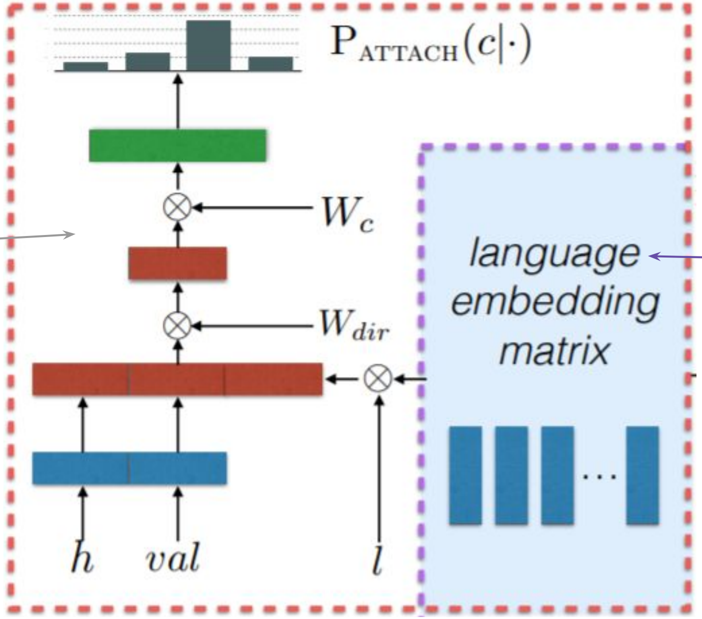


Multilingual grammar induction

Unified models for different languages [Han et al., 2019].

- Capture language similarities by learning *language embeddings*

shared
model parameters



continuous
language indicators