A Generative Model for Parsing Natural Language to Meaning Representations

(Lu, Ng, Lee & Zettlemoyer 2008)

Khaleeq Ahmad

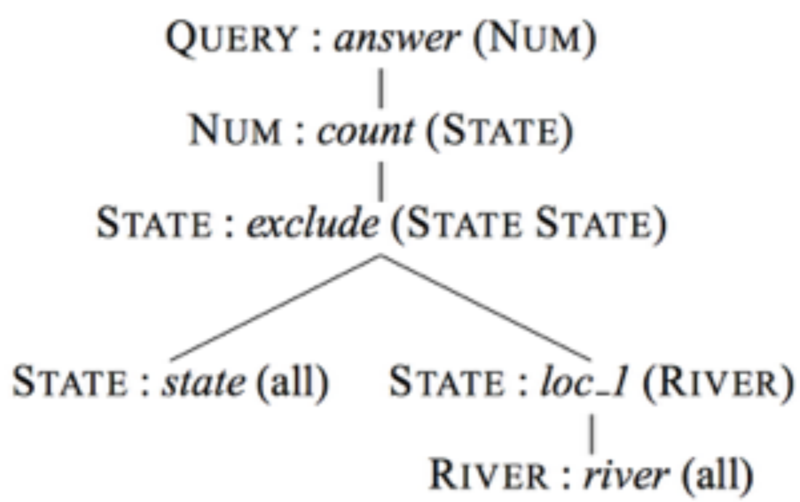
Introduction

- Learn a ***generative model*** which maps sentences with a hierarchical ***meaning representation.***

Sentence
(Natural language)

Meaning
Representation

"How many states do not have rivers?"



Introduction

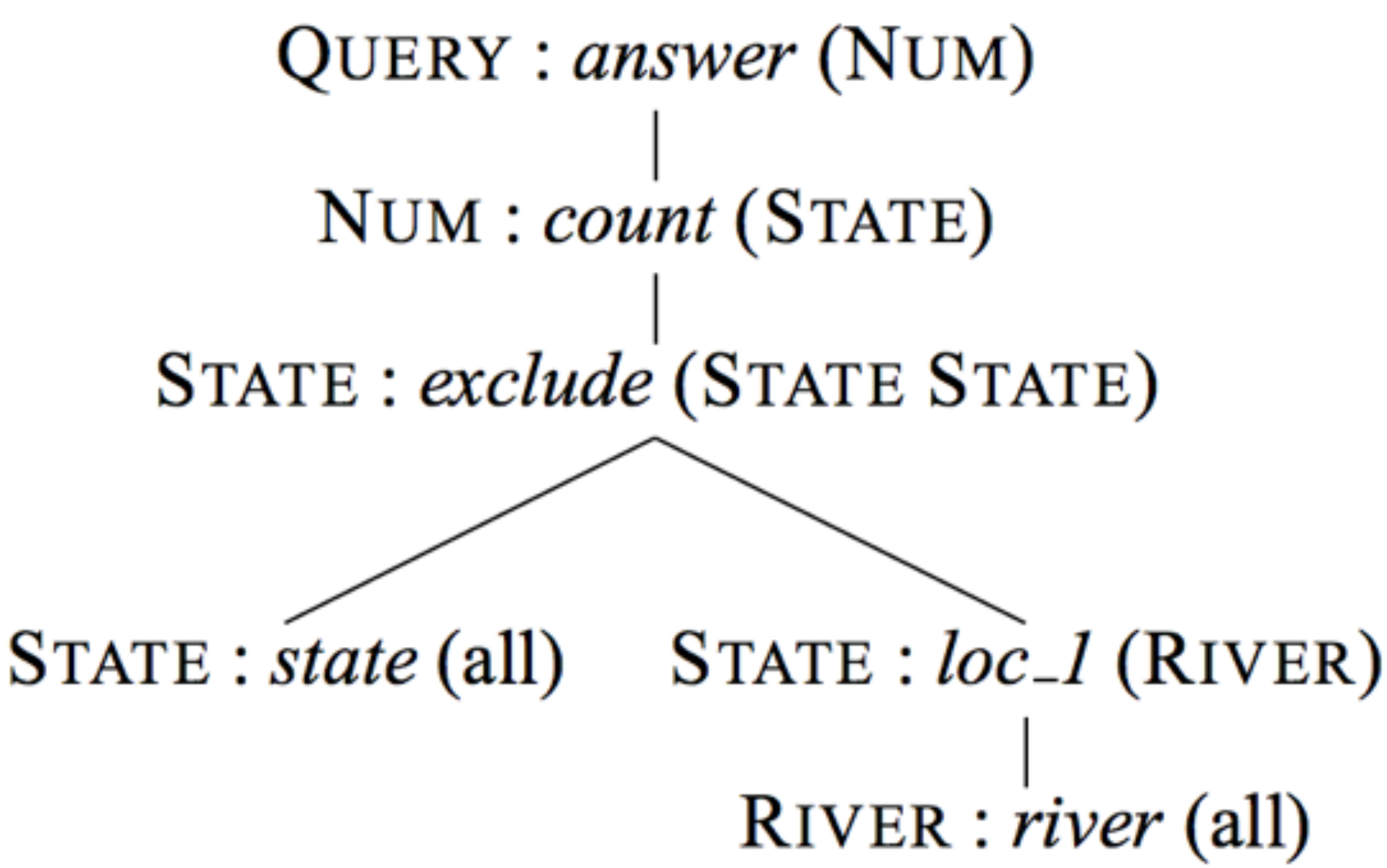
- Learn a ***generative model*** which maps sentences with a hierarchical ***meaning representation.***
- The meaning representation will be in the form of a *hybrid tree*
- To improve on the generative model, will use a reranking algorithm to pick the best tree from a set of top *k* candidates.

Meaning Representations (MR)

- Formal semantic representation written in some ***meaning representation language*** (MRL)

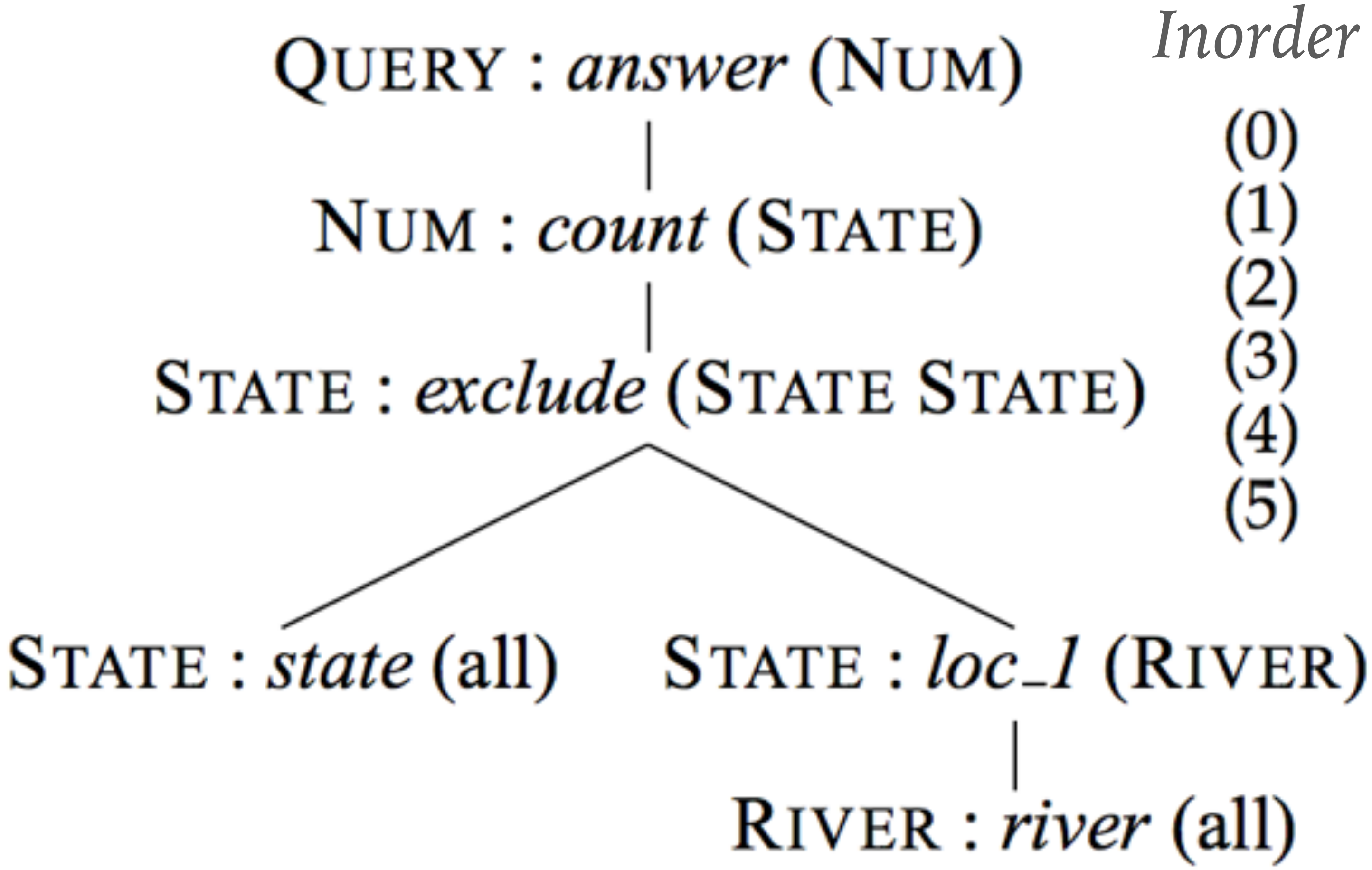
Meaning Representations (MR)

- Formal semantic representation written in some ***meaning representation language*** (MRL)
- E.g. "How many states do not have rivers?"



Meaning Representations (MR)

- Formal semantic representation written in some ***meaning representation language*** (MRL)
- E.g. "How many states do not have rivers?"

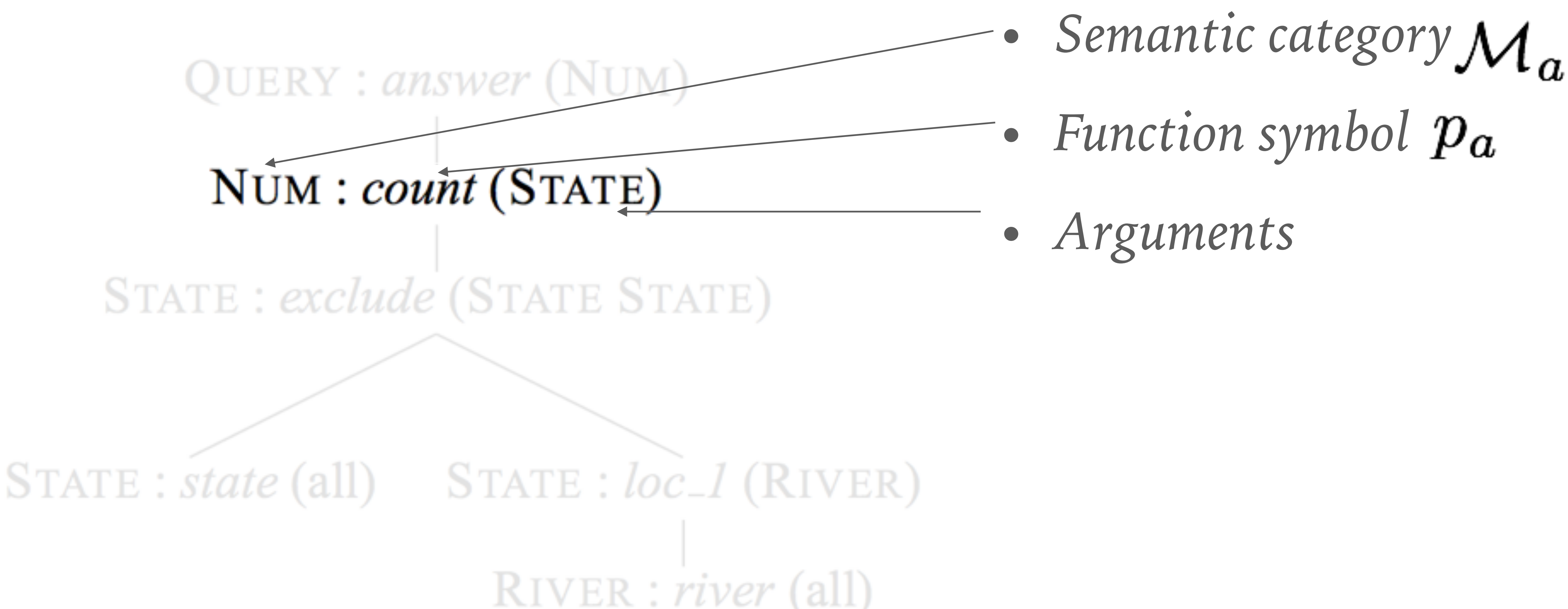


Inorder traversal:

(0)	QUERY	:	*answer* (NUM)
(1)	NUM	:	*count* (STATE)
(2)	STATE	:	*exclude* (STATE_1 STATE_2)
(3)	STATE	:	*state* (all)
(4)	STATE	:	*loc_1* (RIVER)
(5)	RIVER	:	*river* (all)

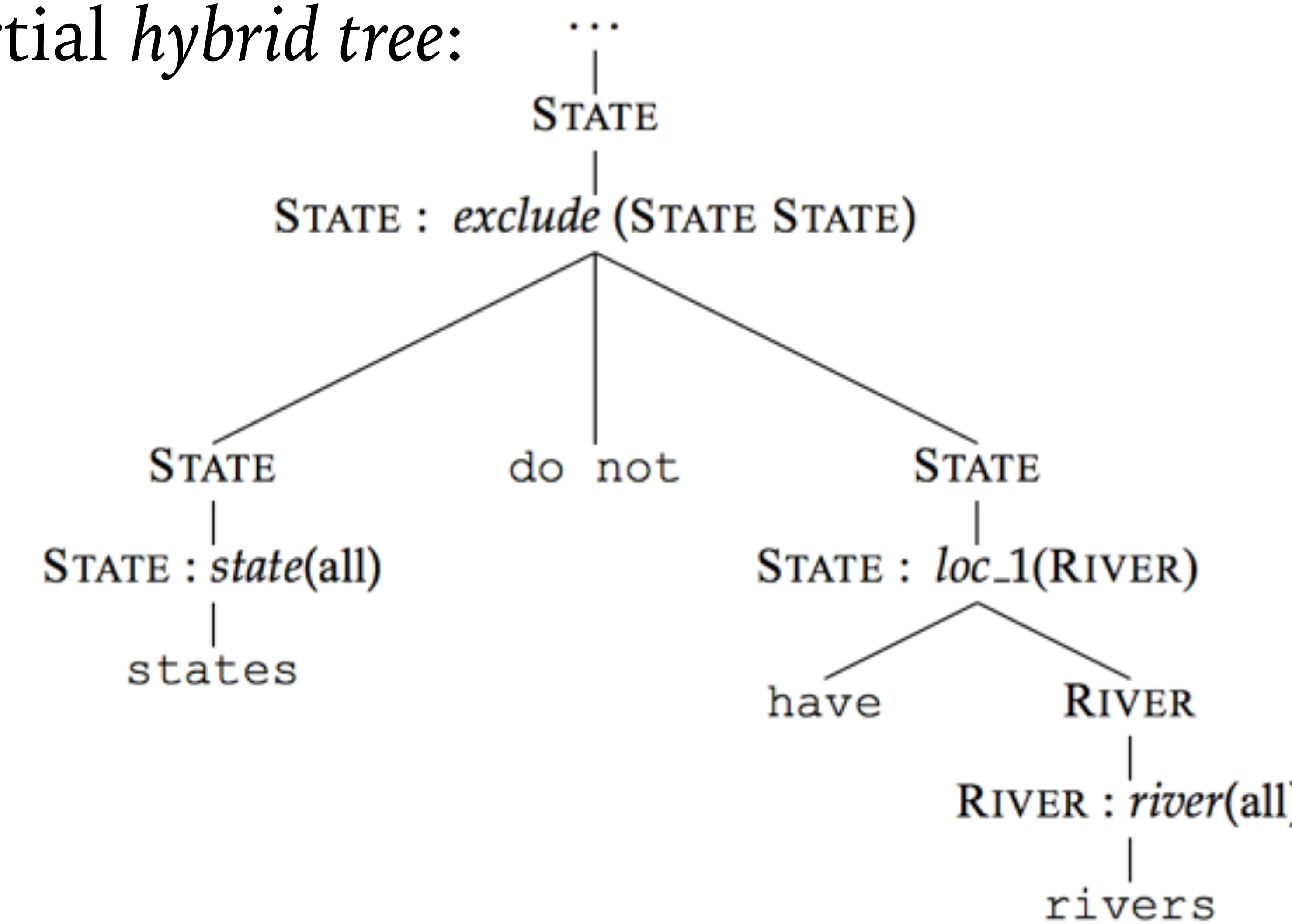
Meaning Representations (MR)

- Formal semantic representation written in some ***meaning representation language*** (MRL)
- Each production consists of:



Generative model

- The aim of the generative model is to simultaneously generate NL sentences and MR structures:
- This is an example partial *hybrid tree*:

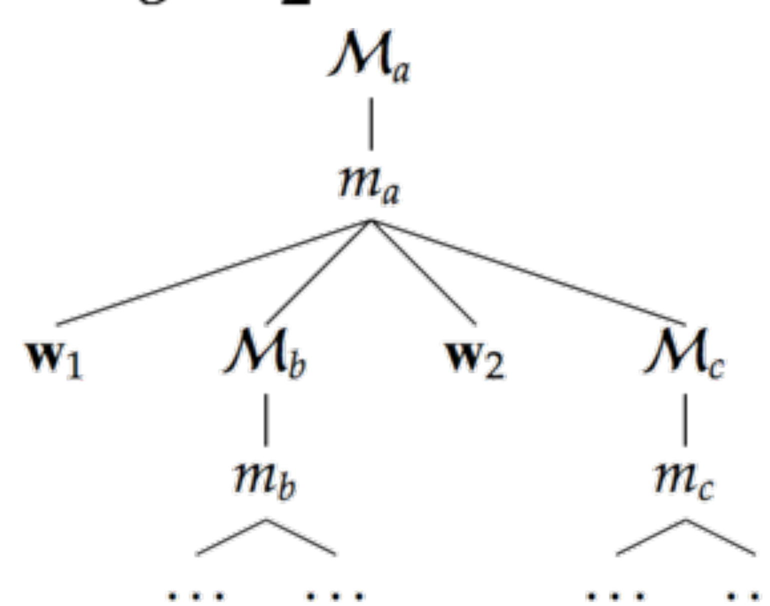


Building a tree

- Sentence: $\widehat{\mathbf{w}} = \mathbf{w}_1 \ldots \mathbf{w}_n$
- MR structures: $\widehat{\mathbf{m}}$
- MR productions: $m_a \quad = \quad \mathcal{M}_a : p_a(\mathcal{M}_b, \mathcal{M}_c)$

➤ Process to generate *hybrid tree* $\mathcal{T}$:

  - ➤ Given semantic category $\mathcal{M}_a$ pick a production m_a
  - ➤ Generate child nodes as *hybrid sequence* $\overline{\mathcal{M}_b \ \mathbf{w}_1 \ \mathcal{M}_c \ \mathbf{w}_2}$



  - ➤ Repeat recursively on new category nodes
  - ➤ On a full tree the leaves will all be sentence tokens

Probability of a tree

- Based on independence assumptions, probability of $\langle \widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T} \rangle$

$$
\begin{aligned}
P(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T}) = P(\mathcal{M}_a) \times P(m_a|\mathcal{M}_a) \times P(\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c}|m_a) \\
\times P(m_b|m_a, \text{arg} = 1) \times P(\ldots|m_b) \\
\times P(m_c|m_a, \text{arg} = 2) \times P(\ldots|m_c)
\end{aligned}
$$

Probability of a tree

- Based on independence assumptions, probability of $\langle \widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T} \rangle$

$$
\begin{aligned}
P(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T}) = P(\mathcal{M}_a) \times P(m_a | \mathcal{M}_a) \times P(\overline{\mathbf{w}_1\ \mathcal{M}_b\ \mathbf{w}_2\ \mathcal{M}_c} | m_a) \\
\times P(m_b | m_a, \mathrm{arg} = 1) \times P(\ldots | m_b) \\
\times P(m_c | m_a, \mathrm{arg} = 2) \times P(\ldots | m_c)
\end{aligned}
$$

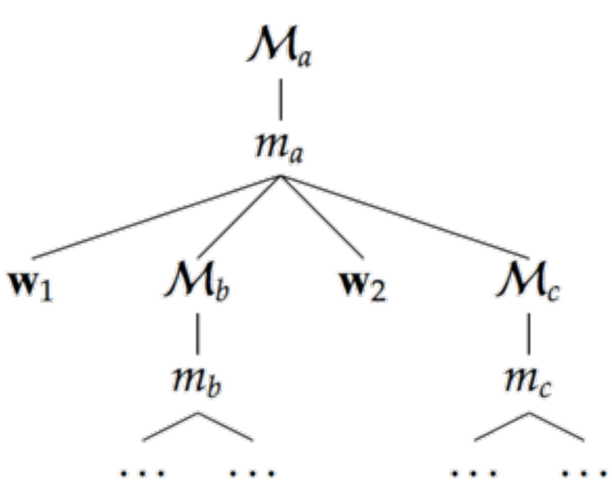
- Possible hybrid patterns:

# RHS	Hybrid Pattern	# Patterns
0	$m \rightarrow \mathbf{w}$	1
1	$m \rightarrow [\mathbf{w}]\mathcal{Y}[\mathbf{w}]$	4
2	$m \rightarrow [\mathbf{w}]\mathcal{Y}[\mathbf{w}]\mathcal{Z}[\mathbf{w}]$	8
	$m \rightarrow [\mathbf{w}]\mathcal{Z}[\mathbf{w}]\mathcal{Y}[\mathbf{w}]$	8

Probability of a tree

- Based on independence assumptions, probability of $\langle \widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T} \rangle$

$$P(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T}) = P(\mathcal{M}_a) \times P(m_a | \mathcal{M}_a) \times P(\overline{\mathbf{w}_1 \ \mathcal{M}_b \ \mathbf{w}_2 \ \mathcal{M}_c} | m_a)$$
$$\times P(m_b | m_a, \mathrm{arg} = 1) \times P(\ldots | m_b)$$
$$\times P(m_c | m_a, \mathrm{arg} = 2) \times P(\ldots | m_c)$$



$$P(\overline{\mathbf{w}_1 \ \mathcal{M}_b \ \mathbf{w}_2 \ \mathcal{M}_c} | m_a) = P(m \to \mathbf{w} \mathcal{Y} \mathbf{w} \mathcal{Z} | m_a) \times P(\mathbf{w}_1 | m_a)$$
$$\times P(\mathcal{M}_b | m_a, \mathbf{w}_1) \times P(\mathbf{w}_2 | m_a, \mathbf{w}_1, \mathcal{M}_b)$$
$$\times P(\mathcal{M}_c | m_a, \mathbf{w}_1, \mathcal{M}_b, \mathbf{w}_2) \times P(\mathrm{END} | m_a, \mathbf{w}_1, \mathcal{M}_b, \mathbf{w}_2, \mathcal{M}_c)$$

Parameter types

- Three categories of parameters:
  - MR model parameters:

    $\sum_{m'} \rho(m'|m_j, \mathrm{arg}=k) = 1$ for all j and $k = 1, 2$.
  - Emission parameters:

    $\sum_t \theta(t|m_j, \Lambda) = 1$ for all j

    where t is a node and Λ is the context.
  - Pattern parameters:

    $\sum_r \phi(r|m_j) = 1$ for all j where r is a hybrid pattern.

Models

- Three models based different context assumptions
  - Model 1 assumes: $\theta(t_k|m_j, \Lambda) = P(t_k|m_j)$
  - Model 2 assumes: $\theta(t_k|m_j, \Lambda) = P(t_k|m_j, t_{k-1})$
  - Model 3 assumes: $\theta(t_k|m_j, \Lambda) = \frac{1}{2} \times \left(P(t_k|m_j) + P(t_k|m_j, t_{k-1})\right)$

Parameter estimation

- MR parameters
  - Simply count productions from the corpus.
  - Then normalise.
- Generative parameters: (Emission and Pattern)
  - Do not know alignment between words and MR
  - Use Expectation Maximisation to re-estimate parameters
    - via Inside-Outside dynamic programming
    - Smoothing

Decoding

- Find MR structure $\widehat{\mathbf{m}}^*$ for a sentence $\widehat{\mathbf{w}}$

$$\widehat{\mathbf{m}}^* = \arg\max_{\widehat{\mathbf{m}}} \sum_{\mathcal{T}} P(\widehat{\mathbf{m}}, \mathcal{T} | \widehat{\mathbf{w}})$$

- This summation is very expensive so instead :
  - Find best approximate

$$= \arg\max_{\widehat{\mathbf{m}}} \max_{\mathcal{T}} P(\widehat{\mathbf{m}}, \mathcal{T} | \widehat{\mathbf{w}}) = \arg\max_{\widehat{\mathbf{m}}} \max_{\mathcal{T}} P(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}, \mathcal{T})$$

    - Used candidate ranking algorithm to find top k and then Viterbi to select best.

Re-ranking and filtering

- Generative model unable to model long range dependencies
- What if wrong candidate is chosen?
- Postprocess with discriminative re-ranking algorithm

Averaged Perceptron

- Three components:
  - GEN function: finds set of candidate trees for sentence.

    Use decoding function to output k hybrid trees.
    For system $k = 50$

  - Reference tree for each training instance

    Run Viterbi on each pair to find best reference

  - Feature function : mapping a tree to a feature vector $\Phi(\mathcal{T})$

Feature vector $\Phi(\mathcal{T})$

- Features

Feature Type	Description	Example
1. Hybrid Rule	A MR production and its child hybrid form	f_1 : STATE : *loc_1*(RIVER) → `have` RIVER
2. Expanded Hybrid Rule	A MR production and its child hybrid form expanded	f_2 : STATE : *loc_1*(RIVER) → ⟨`have`, RIVER : *river*(all)⟩
3. Long-range Unigram	A MR production and a NL word appearing below in tree	f_3 : STATE : *exclude*(STATE STATE) → `rivers`
4. Grandchild Unigram	A MR production and its grandchild NL word	f_4 : STATE : *loc_1*(RIVER) → `rivers`
5. Two Level Unigram	A MR production, its parent production, and its child NL word	f_5 : ⟨RIVER : *river*(all), STATE : *loc_1*(RIVER)⟩ → `rivers`
6. Model Log-Probability	Logarithm of base model's joint probability	$\log(\hat{P}(\mathbf{w}, \mathbf{m}, \mathcal{T}))$.

  - 1-5: Indicator features
  - 6: Real value
- Learns a weight vector $\mathbf{w}$ for each $\Phi(\mathcal{T})$
- Aggregate $\mathbf{w}$ into a score for each $\mathcal{T}$
- Pick best candidate tree
- Separating plane to optimise f-measure

Evaluation training

- Two corpora:
  - GEOQUERY: MR defined by Prolog-based language (880)
  - ROBOCUP: MR defined by robot coaching language (300)

- Training:
  - Train 100 iterations of EM on Model I
  - Use these to initialise Model II, train further 100 its
  - Model III: Interpolation

  - Reranking: Run perceptron for 10 iterations

Measure correctness

- Correctness:
  - GEOQUERY:
    MR is correct when it retrieves identical results to reference MR.
  - ROBOCUP:
    MR is correct when it has same string representation as the reference MR.

Evaluation metrics

- Precision

  % answered sentences which are correct / All sentences

- Recall

  % correctly answered sentences / All sentences

- F-measure

  Harmonic mean of *Precision* and *Recall*

Evaluation of Models I, II and III

Model	GEOQUERY (880)			ROBOCUP (300)		
	Prec.	*Rec.*	*F*	*Prec.*	*Rec.*	*F*
I	81.3	77.1	79.1	71.1	**64.0**	67.4
II	**89.0**	76.0	82.0	**82.4**	57.7	**67.8**
III	86.2	**81.8**	**84.0**	70.4	63.3	66.7
I+R	87.5	80.5	83.8	79.1	67.0	72.6
II+R	**93.2**	73.6	82.3	**88.4**	56.0	68.6
III+R	89.3	**81.5**	**85.2**	82.5	**67.7**	**74.4**

Compared to other models

System	GEOQUERY (880)			ROBOCUP (300)		
	Prec.	*Rec.*	*F*	*Prec.*	*Rec.*	*F*
SILT	89.0	54.1	67.3	83.9	50.7	63.2
WASP	87.2	74.8	80.5	**88.9**	61.9	73.0
KRISP	**93.3**	71.7	81.1	85.2	61.9	71.7
Model III+R	89.3	**81.5**	**85.2**	82.5	**67.7**	**74.4**

Performance in other languages

System	English			Spanish		
	Prec.	*Rec.*	*F*	*Prec.*	*Rec.*	*F*
WASP	95.42	70.00	80.76	91.99	72.40	81.03
Model III+R	91.46	72.80	**81.07**	95.19	79.20	**86.46**

System	Japanese			Turkish		
	Prec.	*Rec.*	*F*	*Prec.*	*Rec.*	*F*
WASP	91.98	74.40	**82.86**	96.96	62.40	75.93
Model III+R	87.56	76.00	81.37	93.82	66.80	**78.04**

Conclusion

- New generative model that simultaneously produces both NL sentences and their corresponding MR structures.
- This is combined with dynamic algorithms for training and re-ranking to provide best candidate.
- Has state-of-art performance, outperforming other similar models when tested on two corpora.
- System is also language-independent.
- Would be interesting to see future work on generating a sentence from an MR structure.

References

- W. Lu, H. T. Ng, W. S. Lee, L. S. Zettlemoyer. "A Generative Model for Parsing Natural Language to Meaning Representations". *Conference on Empirical Methods on Natural Language Processing*, 2008.