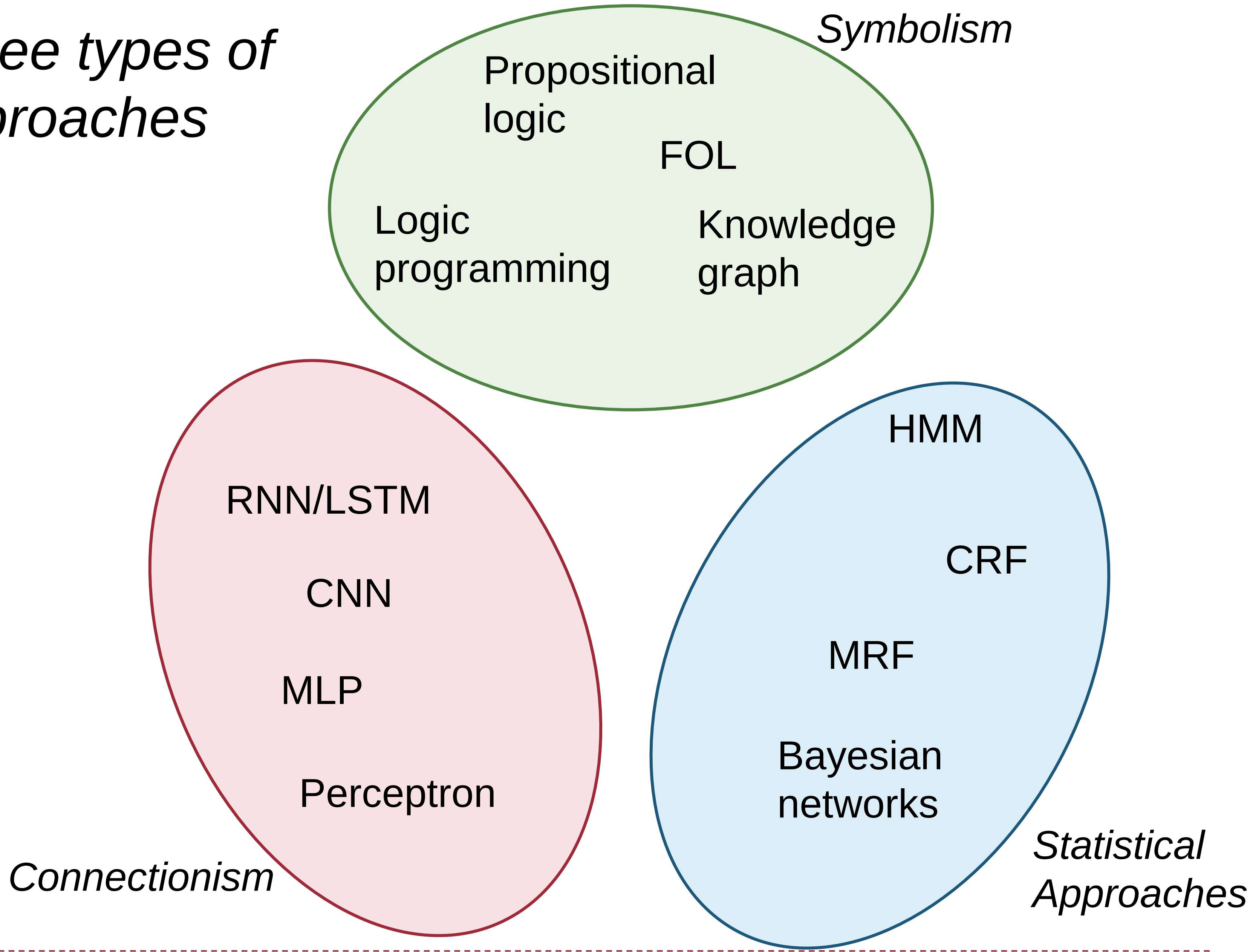
Neuralizing Symbolic Approaches to NLP

Kewei Tu

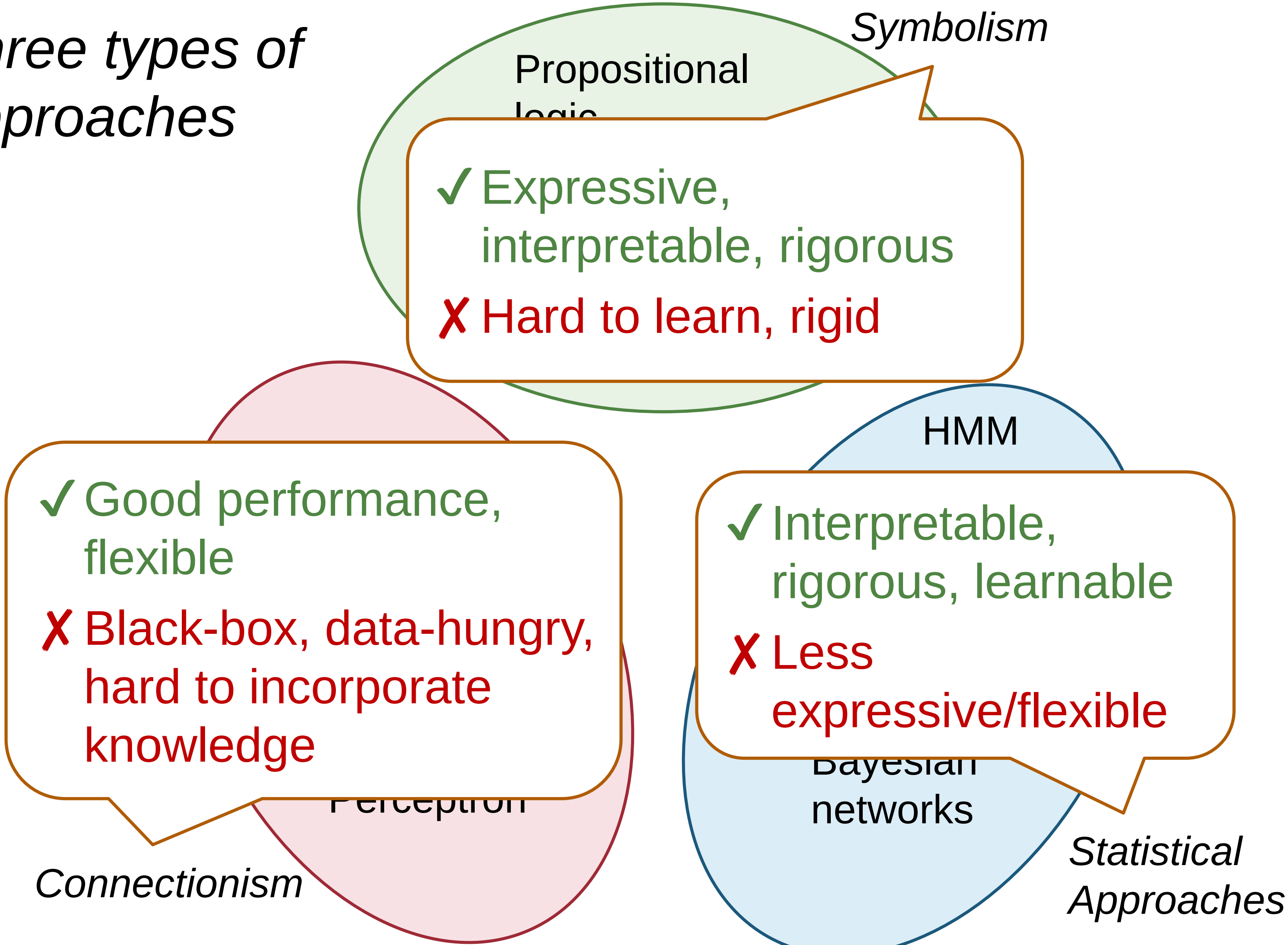
ShanghaiTech University

上海科技大学
ShanghaiTech University

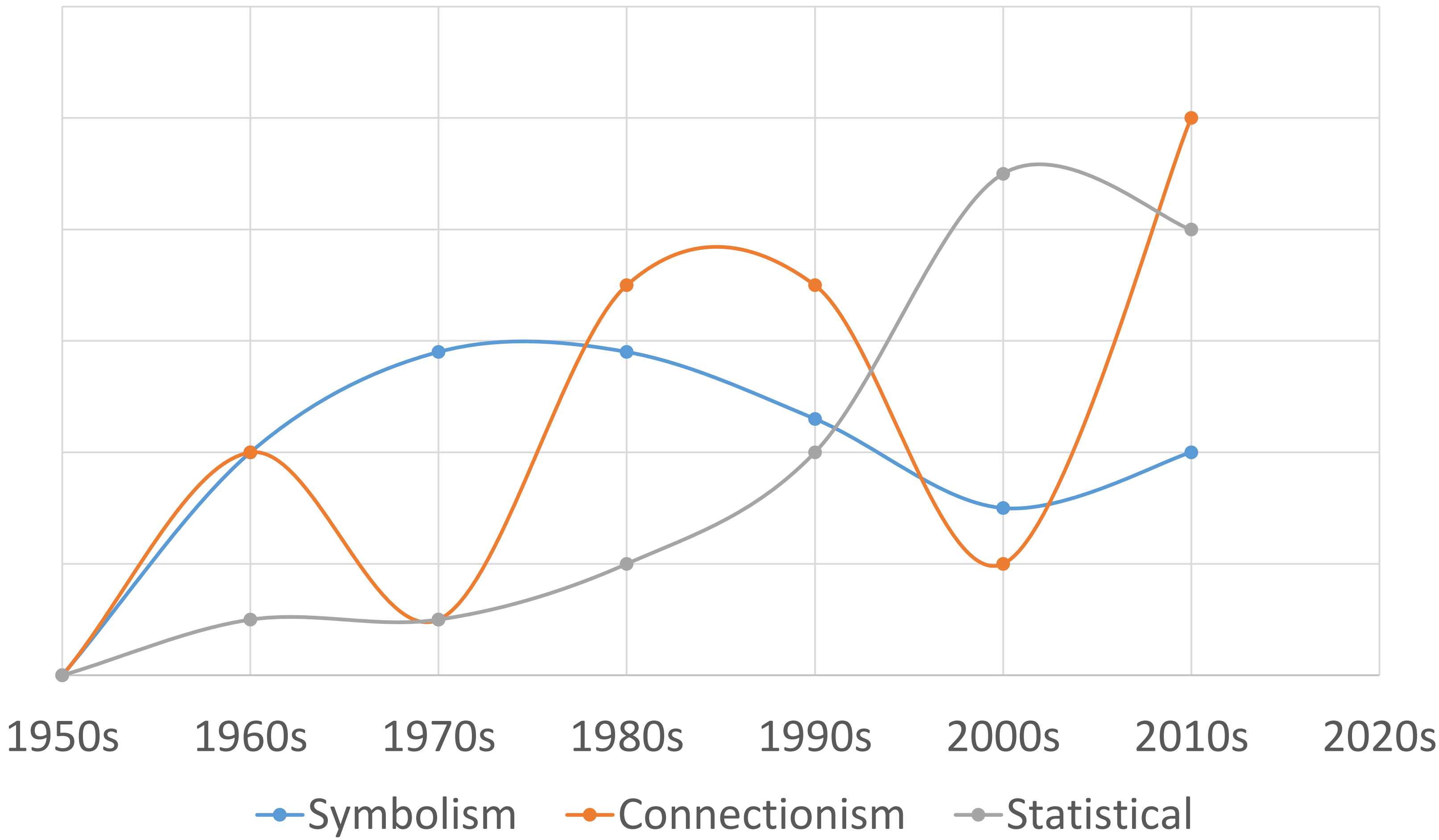
Three types of approaches



Three types of approaches

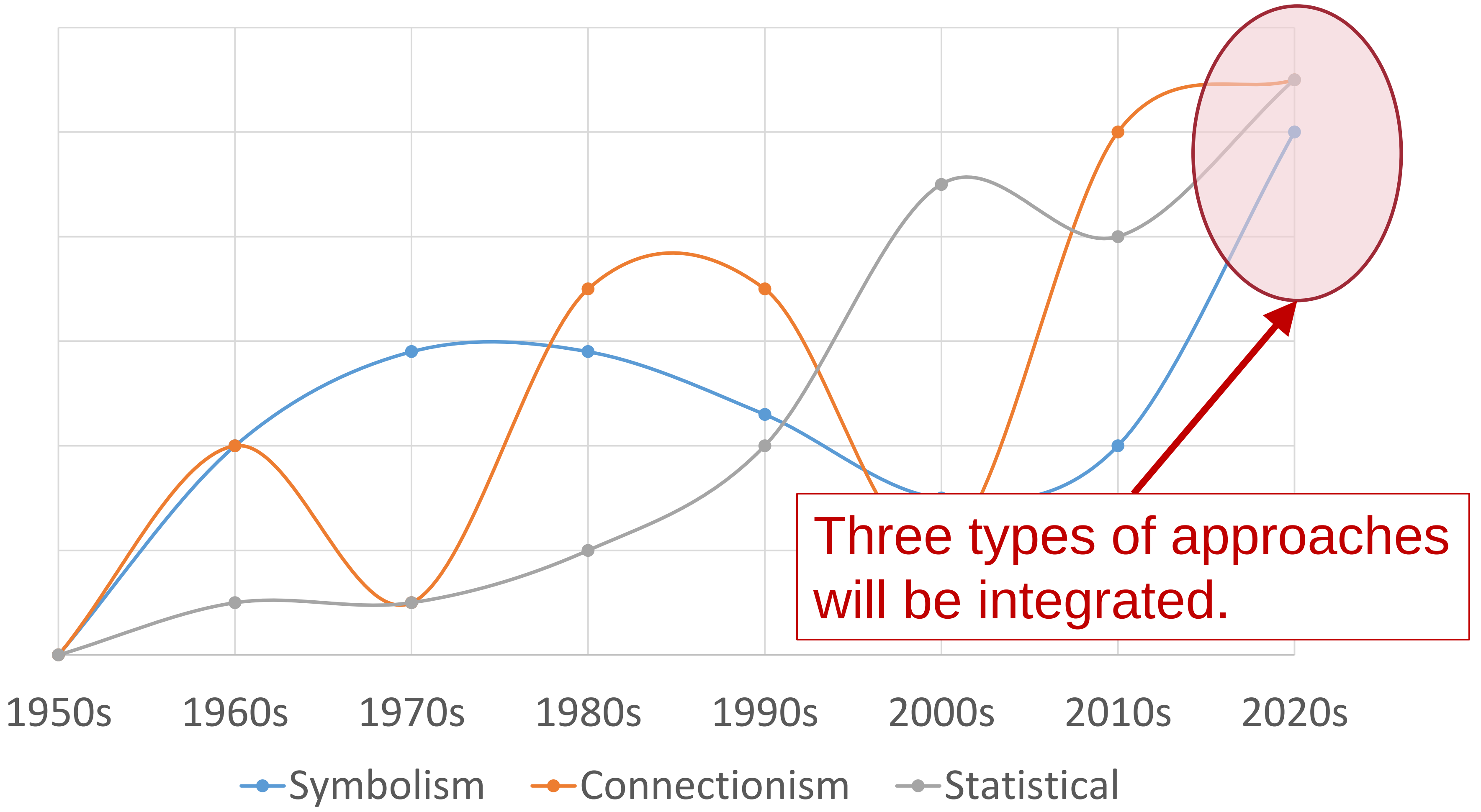


Trends



1950s
1960s
1970s
1980s
1990s
2000s
2010s
2020s
Symbolism
Connectionism
Statistical

Trends


Three types of approaches will be integrated.
1950s
1960s
1970s
1980s
1990s
2000s
2010s
2020s
Symbolism
Connectionism
Statistical

Today's topic

- Neuralizing Symbolic Approaches to NLP
  - Turning symbolic systems to neural networks
  - Learning symbolic systems using neural networks

Symbolism

✓ Expressive, interpretable, rigorous

✗ Hard to learn, rigid

✓ Good performance, flexible

✗ Black-box, data-hungry, hard to incorporate knowledge

Connectionism

Outline

- Symbolism vs. Connectionism
- Turning symbolic systems to neural networks
  - Chengyue Jiang, Yinggong Zhao, Shanbo Chu, Libin Shen, and Kewei Tu, "Cold-start and Interpretability: Turning Regular Expressions into Trainable Recurrent Neural Networks", EMNLP 2020.
- Learning symbolic systems using neural networks

Regular Expressions (RE)

- One of the most representative and useful forms of symbolic rules
- Widely used in practice: text classification, slot filling, etc.

Label	[*distance*]
RE	*$*(how (far | long) | distance) $**
Matched Text	⟨BOS⟩ tell me **how far** is oakland airport from downtown ⟨EOS⟩

Regular Expressions (RE)

- Pros
  - Highly interpretable
  - Support fine-grained diagnosis and manipulation
    - Easy to add/delete/revise rules to quickly adapt to changes in task specification
  - No need for training
    - Hence no need for data annotation, less computational cost
    - Good for cold-start scenarios
- Cons
  - Rely on human experts to write
  - Often: high precision but low recall
  - Cannot evolve by training on labeled data when available
    - Underperform neural approaches in rich-resource scenarios

Our Idea

- Convert a RE to a new form of recurrent neural networks
  - Roughly equivalent to RE
    - ✓ Can still be used in cold-start scenarios
  - Trainable on labeled data
    - ✓ Can outperform REs and compete with neural approaches in rich-resource scenarios
  - Can be converted back to RE
    - ✓ Possibility of fine-grained manipulation

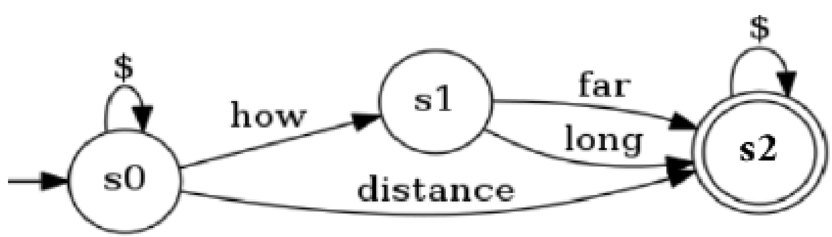
- Let's start with classification…

Step 1. RE to Finite Automaton (FA)

- Any RE can be converted into a FA that expresses the same language

Label	[*distance*]
RE	*$*(how (far | long) | distance) $**

FA



FA parameters

- Binary transition tensor:

  $$T \in \mathbb{R}^{V \times K \times K}$$

- Binary start vector:

  $$\alpha_0 \in \mathbb{R}^{K}$$

- Binary final vector:

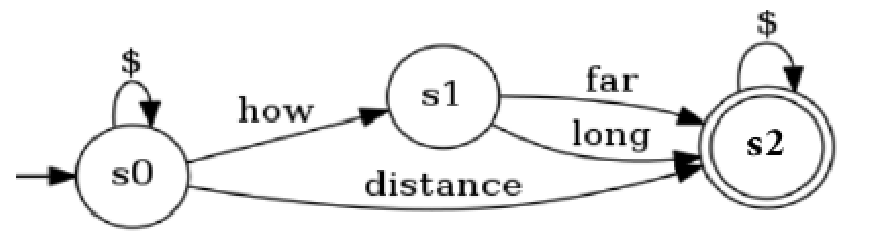
  $$\alpha_\infty \in \mathbb{R}^{K}$$

V: vocabulary size

K: state number

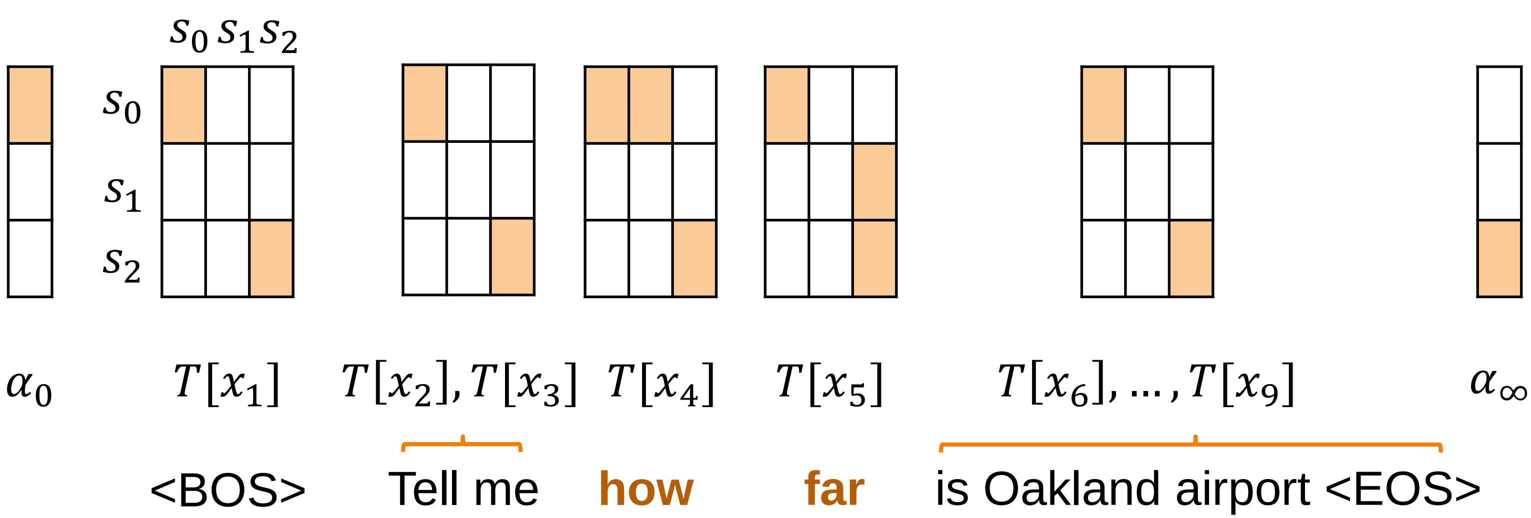
Step 2. FA as Recurrent Neural Network (RNN)

- Score of a FA accepting a sentence can be calculated using the forward algorithm



Forward score:

$$\boldsymbol{\alpha}_0^T \cdot \left(\prod_{i=1}^{N} \boldsymbol{T}[x_i]\right) \cdot \boldsymbol{\alpha}_\infty$$



Step 2. FA as Recurrent Neural Network (RNN)

- The computation can be rewritten into a recurrent form

$$\boldsymbol{\alpha}_0^T \cdot \left(\prod_{i=1}^{N} \boldsymbol{T}[x_i] \right) \cdot \boldsymbol{\alpha}_\infty$$

$$\Downarrow$$

$$\boldsymbol{h}_{\mathbf{0}} = \boldsymbol{\alpha}_0^T$$

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} \cdot \boldsymbol{T}[x_t],\ 1 \leq t \leq N \quad \text{(recurrent step)}$$

$$\mathcal{B}_{\text{forward}}(\mathcal{A}, \boldsymbol{x}) = \boldsymbol{h}_N \cdot \boldsymbol{\alpha}_\infty$$

Step 3. Decomposing the Parameter Tensor

- Goal: reduce the computational complexity to match that of traditional RNN

Tensor Rank Decomposition

$$\boldsymbol{T} \in \mathbb{R}^{V\times K\times K} \Rightarrow \boldsymbol{E}_{\mathcal{R}} \in \mathbb{R}^{V\times r}, \boldsymbol{D}_1 \in \mathbb{R}^{K\times r}, \boldsymbol{D}_2 \in \mathbb{R}^{K\times r}$$

(word embedding) (state embeddings)

- Now the recurrent step becomes:

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} \cdot \boldsymbol{T}[x_t] \Rightarrow \begin{aligned} \boldsymbol{v}_t &= \boldsymbol{E}_{\mathcal{R}}(x_t) \\ \boldsymbol{a} &= (\boldsymbol{h}_{t-1} \cdot \boldsymbol{D}_1) \circ \boldsymbol{v}_t \\ \boldsymbol{h}_t &= \boldsymbol{a} \cdot \boldsymbol{D}_2^T \end{aligned}$$

Step 4. Integrating Pretrained Word Embedding

- Goal: bringing external lexical knowledge into our model
- Method:
  - Approximate $\boldsymbol{E}_{\mathcal{R}}$ with $\boldsymbol{E}_w\boldsymbol{G}$

    initialized with $\boldsymbol{E}_w^{\dagger}\boldsymbol{E}_{\mathcal{R}}$

    $\boldsymbol{E}_w^{\dagger}$ is the pseudo-inverse of $\boldsymbol{E}_w$

    external word embedding
  - Interpolate $\boldsymbol{E}_{\mathcal{R}}$ and $\boldsymbol{E}_w\boldsymbol{G}$
  - The recurrent step becomes:

$$
\begin{aligned}
\boldsymbol{v}_t &= \boldsymbol{E}_{\mathcal{R}}(x_t)\\
\boldsymbol{a} &= (\boldsymbol{h}_{t-1} \cdot \boldsymbol{D}_1) \circ \boldsymbol{v}_t\\
\boldsymbol{h}_t &= \boldsymbol{a} \cdot \boldsymbol{D}_2^T
\end{aligned}
\quad\Rightarrow\quad
\begin{aligned}
\boldsymbol{v}_t &= \boldsymbol{E}_{\mathcal{R}}(x_t) \quad \boldsymbol{u}_t = \boldsymbol{E}_w(x_t)\\
\boldsymbol{z}_t &= \beta \boldsymbol{v}_t + (1-\beta)\boldsymbol{u}_t\boldsymbol{G}\\
\boldsymbol{a} &= (\boldsymbol{h}_{t-1} \cdot \boldsymbol{D}_1) \circ \boldsymbol{z}_t\\
\boldsymbol{h}_t &= \boldsymbol{a} \cdot \boldsymbol{D}_2^T
\end{aligned}
$$

FA-RNN

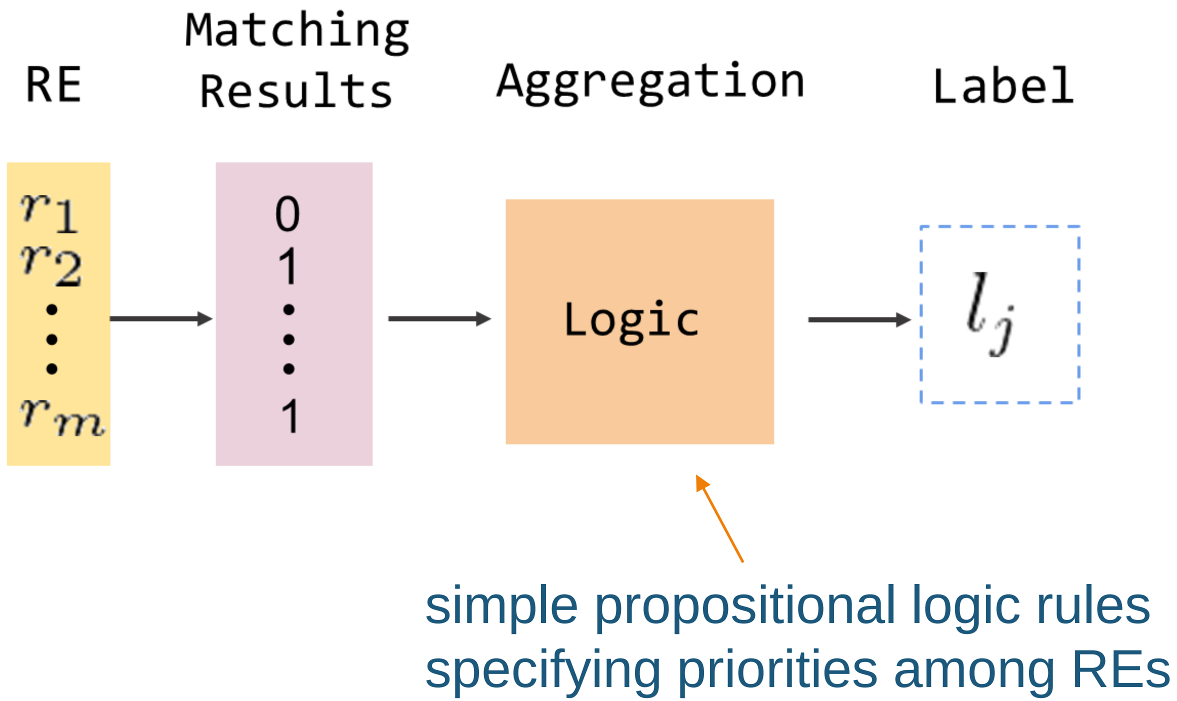
FA-RNN Extensions

- Gated extension
  - Add forget gate and reset gate like in GRU
  - Initialize parameters to make the gates inactive initially

- Combine two FA-RNNs of opposite directions
  - Create a left-to-right FA-RNN from the RE
  - Create a right-to-left FA-RNN from the reversed RE
  - Output the average score of the two FA-RNNs

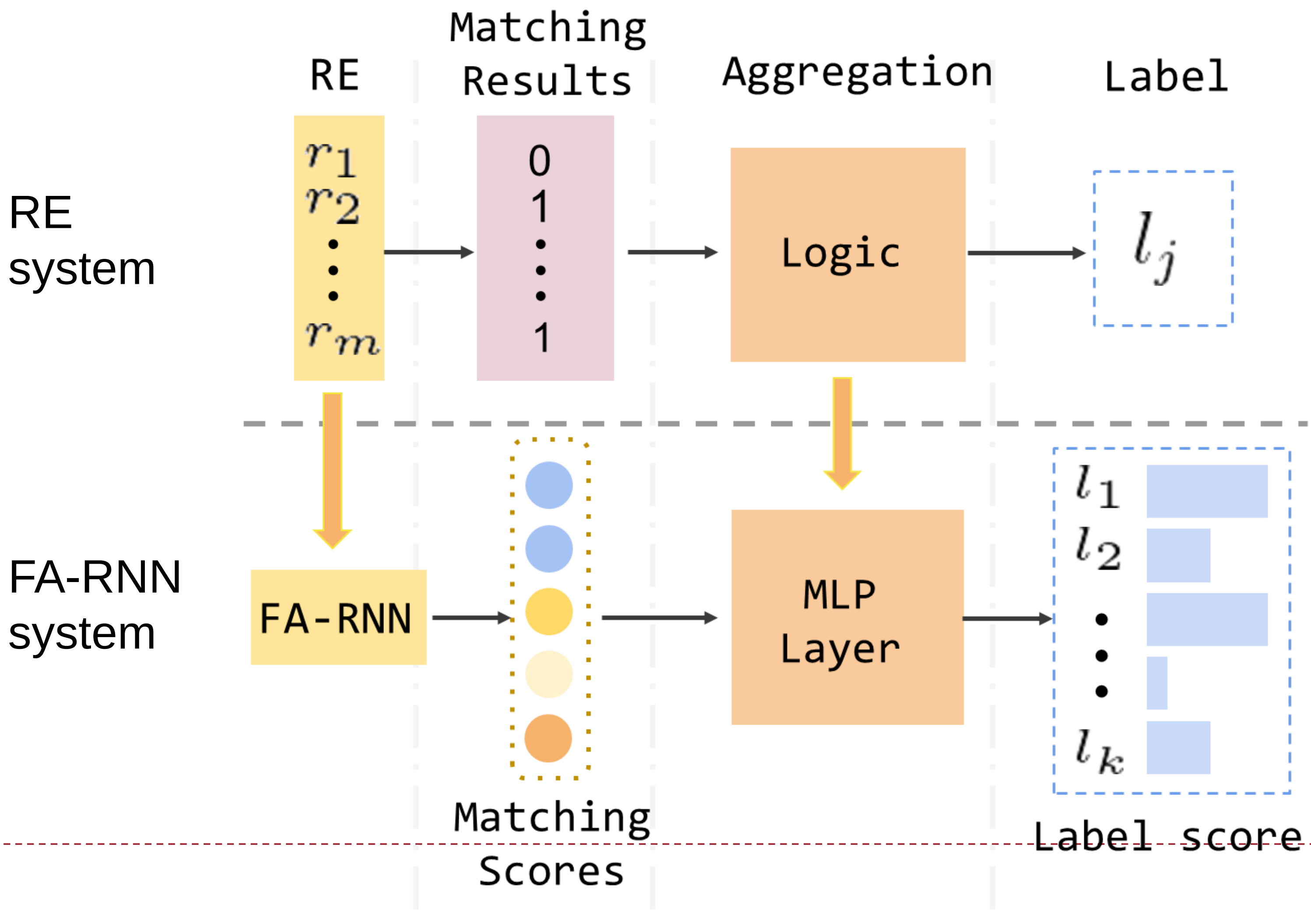
Text classification

- An RE system for text classification:
  - Aggregating results from multiple REs to form a prediction



Text classification

- From a RE system to a FA-RNN system

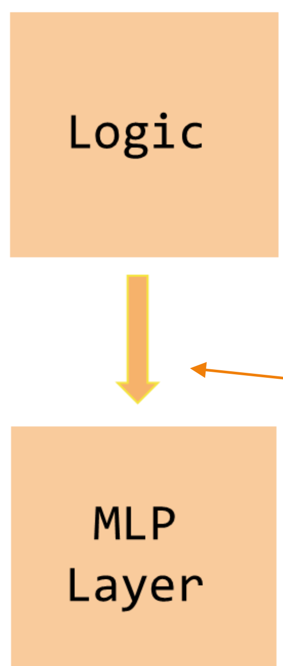


Text classification

- From a RE system to a FA-RNN system

Aggregation

RE system

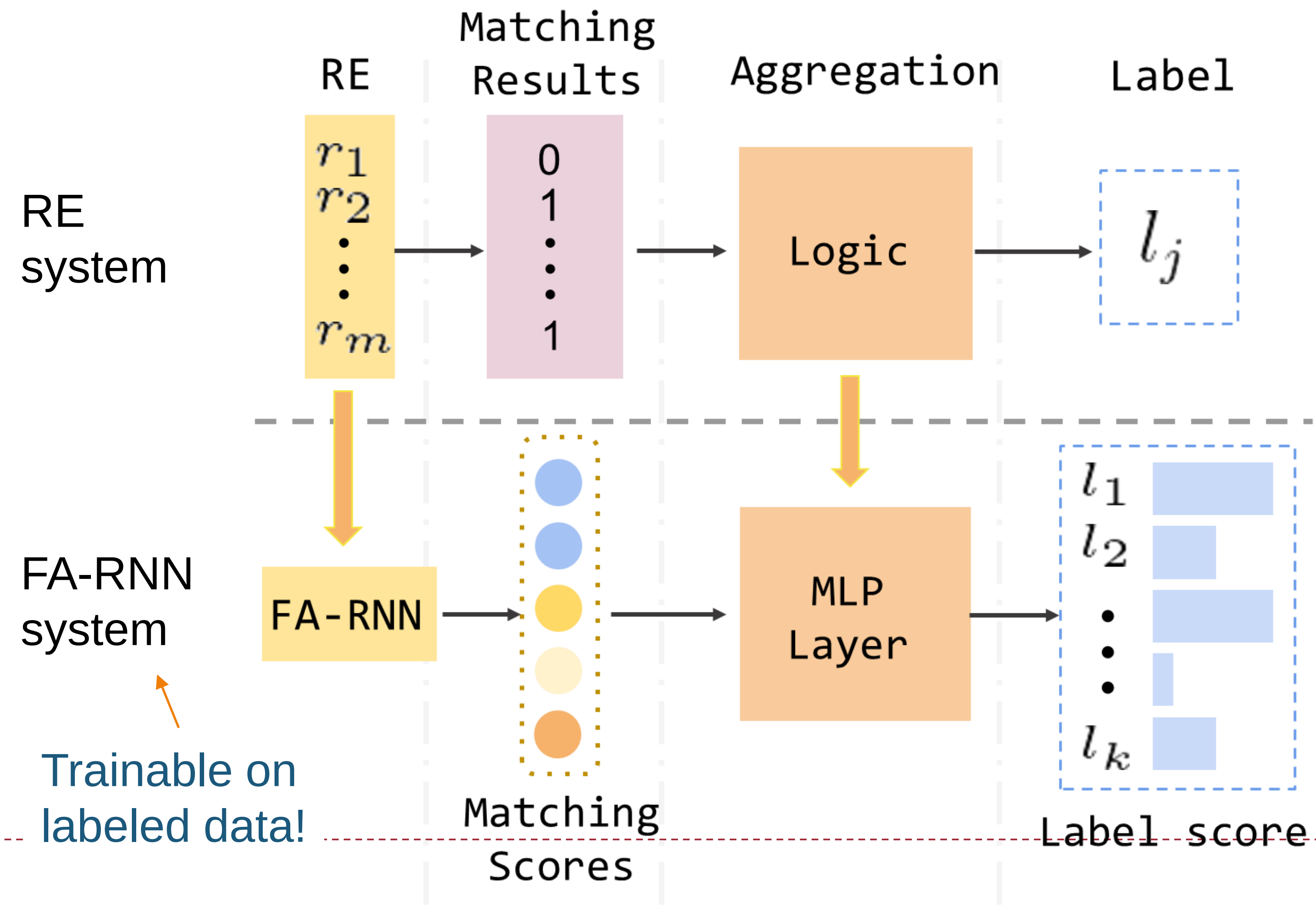


FA-RNN system

Logic	Soft Logic
$\neg A$	$1 - a$
$A \vee B$	$\min(1, a + b)$
$A \wedge B$	$\max(0, a + b - 1)$

Text classification

- From a RE system to a FA-RNN system



Experiments

- Three intent classification datasets:
  - ATIS, QC (TREC-6), SMS
- Baselines
  - Bi-RNN/GRU/LSTM, CNN, DAN
  - RE-enhanced NN (+i, +o, +io) [Luo et al., 2016]
  - Knowledge Distillation (+kd, +pr) [Hinton et al,.2015; Hu et al,. 2016]

Experiments – Zero-Shot

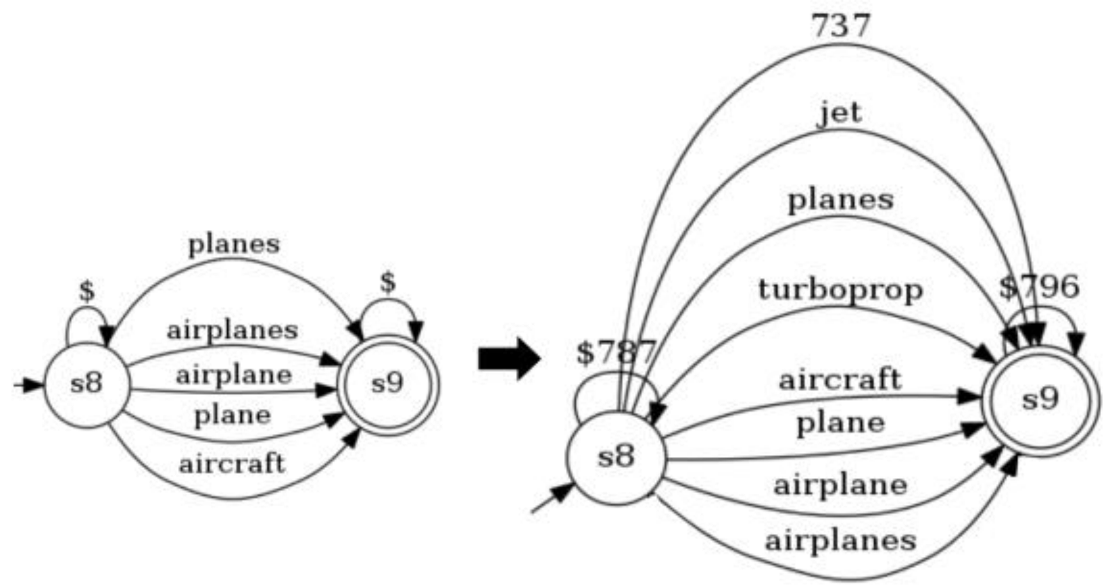
	ATIS	QC	SMS
RE system	87.01	64.40	93.20
FA-RNN	86.53	61.95	93.00
FA-GRU	86.81	62.90	93.20
BiFA-RNN	88.10	62.90	93.00
BiFA-GRU	88.63	62.90	93.20
BiGRU+*i*	1.34	18.75	11.90
BiGRU+*o*	30.74	27.50	30.40
BiGRU+*io*	38.69	25.70	73.25
BiGRU+*pr*	9.94	17.70	53.00
BiGRU+*kd*	9.94	17.70	53.00

Experiments – Low-Resource and Full Training

	ATIS (26-class)			QC (6-class)			SMS (2-class)		
	1%	10%	100%	1%	10%	100%	1%	10%	100%
FA-RNN	90.43	90.79	96.52	**67.75**	79.6	91.3	93.1	96.75	98.8
FA-GRU	88.94	**90.85**	96.61	66.2	80.7	91.85	**94.25**	**96.8**	**99.2**
BiFA-RNN	89.31	**90.85**	96.72	57.65	81.5	91.55	91.7	96.7	99
BiFA-GRU	**90.62**	90.26	96.64	64.15	**82.8**	**92.4**	93.9	96.75	98.8
CNN	71.61	86.09	94.74	50.9	74.9	89.25	89.85	95.9	98.8
DAN	71.02	83.68	90.4	47.25	65.4	77.8	89.9	93.7	98.6
RNN	70.91	75.17	91.55	22.4	67.9	85	85.1	89.85	97.75
LSTM	69.37	78.14	95.72	40.45	75.75	90	86.2	95.75	97.85
GRU	70.72	88.52	96.3	42.35	79.75	91.2	86.15	95.55	98.05
BiRNN	70.72	79.98	93.39	49.35	75.95	87.35	86.75	94.9	97.8
BiLSTM	70.77	87.12	96.25	55.95	76.75	90.95	92.15	95.8	97.7
BiGRU	70.69	88.35	96.75	62.7	80.05	91.5	89.6	95.95	98.4
BiGRU +i	82.84	90.01	96.56	66.3	80.25	92	90.95	96.75	98.55
BiGRU +o	80.21	89.22	96.33	60.15	80.2	91.7	90.6	95.95	98.4
BiGRU +io	82.61	89.95	95.46	65.05	79.65	90.7	93.85	96.75	98.25
BiGRU +pr	72.4	88.89	96.5	61.6	80.45	91.85	90.9	96.05	98.45
BiGRU +kd	73.38	88.86	**96.75**	62.65	80.3	91.25	87.65	96	98.55

Conversion Back to RE

- From a FA-RNN, we can recover a WFA tensor from the model parameters
- The WFA tensor can be rounded to a 0/1 tensor, resulting in a FA and hence a RE
- Extracted RE vs. original RE
  - ATIS: +0.45%
  - QC: +9.2%
  - SMS: -1.2%



Outline

- Symbolism vs. Connectionism
- Turning symbolic systems to neural networks
  - Chengyue Jiang, Zijian Jin, and Kewei Tu, "Neuralizing Regular Expressions for Slot Filling", EMNLP 2021.
- Learning symbolic systems using neural networks

RE for slot filling

- Slot filling

show me flights from (san francisco) to dallas

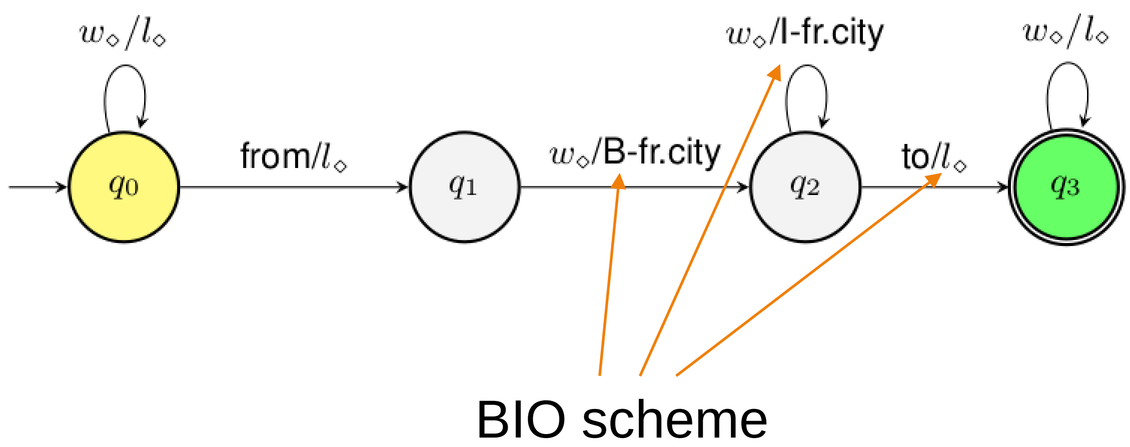
fr.city

- Regular expression to catch fr.city:

$$w_{\diamond}^{*} \quad \mathit{from} \quad \underbrace{[w_{\diamond}^{*}]\langle \mathit{fr.city} \rangle}_{\text{capturing group}} \quad \mathit{to} \quad w_{\diamond}^{*}$$

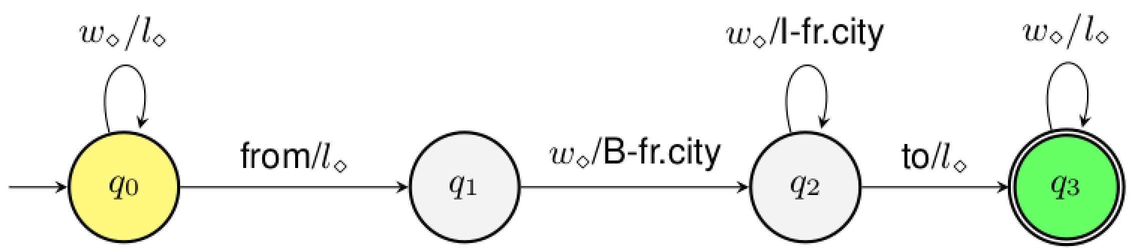
Step 1. RE → Finite State Transducer (FST)

$$w_\diamond^* \;\; \textit{from} \;\; [w_\diamond^*]\langle \textit{fr.city} \rangle \;\; \textit{to} \;\; w_\diamond^*$$



Step 1. RE → Finite State Transducer (FST)

$$w_\diamond^* \quad \textit{from} \quad [w_\diamond^*]\langle \textit{fr.city} \rangle \quad \textit{to} \quad w_\diamond^*$$

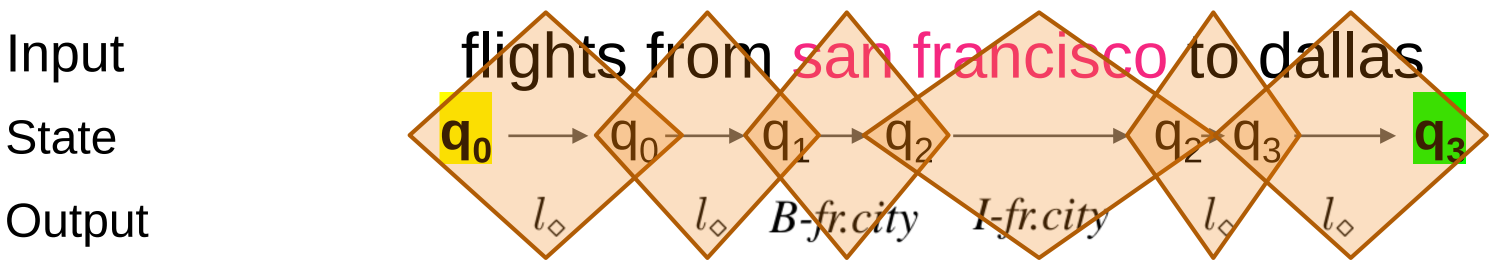


Input	flights from san francisco to dallas
State	$\mathbf{q_0} \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \longrightarrow q_2 \rightarrow q_3 \rightarrow \mathbf{q_3}$
Output	$l_\diamond$ $l_\diamond$ $B\text{-}fr.city$ $I\text{-}fr.city$ $l_\diamond$ $l_\diamond$

Step 1. RE → Finite State Transducer (FST)

- FST parameters
  - Transition tensor $\boldsymbol{T}_{\Omega} \in \mathbb{R}^{V \times L \times K \times K}$
  - Start & final vectors $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{R}^{K}$

Score $\boldsymbol{\mu}(q_0) \times \boldsymbol{T}_{\Omega}(\text{"flight"}, l_{\diamond}, q_0, q_0) \times \cdots\cdots \times \boldsymbol{\nu}(q_3)$



Step 2. FST as BiRNN

- FST inference
  - Given an input sequence, find the highest-scoring output sequence
    - Need to <u>sum out</u> the state sequence & <u>optimize</u> the output sequence
    - *NP-hard!*
  - Given an input sequence, find the highest-scoring output label at each position

1. Compute forward scores $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} \cdot \boldsymbol{T}'_\Omega[x_t]$

   Sum out label dimension of $\boldsymbol{T}_\Omega$

2. Compute backward scores $\boldsymbol{\beta}_{t-1} = \boldsymbol{\beta}_t \cdot \boldsymbol{T}'_\Omega[x_t]^T$
3. Compute label scores at each position

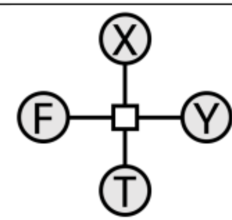
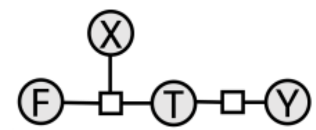
$$(\boldsymbol{c}_t)_k = (\boldsymbol{\alpha}_{t-1})_i \, (\boldsymbol{T}_\Omega[x_t])_{kij} \, (\boldsymbol{\beta}_t)_j$$

Einsum notation

A form of Bidirectional RNN

Step 3. FST → iFST

- Independent FST (iFST)
  - Each label (Y) is independent of the input (X) and source state (F) given the target state (T)

	FST	i-FST
factor graph		
parameter	$\boldsymbol{T}_\Omega \in \mathbb{R}^{V\times L\times K\times K}$	$\boldsymbol{T} \in \mathbb{R}^{V\times K\times K}$ $\boldsymbol{O} \in \mathbb{R}^{L\times K}$

Step 3. FST → iFST

- Independent FST (iFST)
  - Each label (Y) is independent of the input (X) and source state (F) given the target state (T)
  - Inference

1. Forward

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} \cdot \boldsymbol{T}'_\Omega[x_t] \quad \Rightarrow \quad \boldsymbol{\alpha}_t = (\boldsymbol{\alpha}_{t-1} \cdot \boldsymbol{T}[x_t]) \circ \boldsymbol{o}^T$$

2. Backward

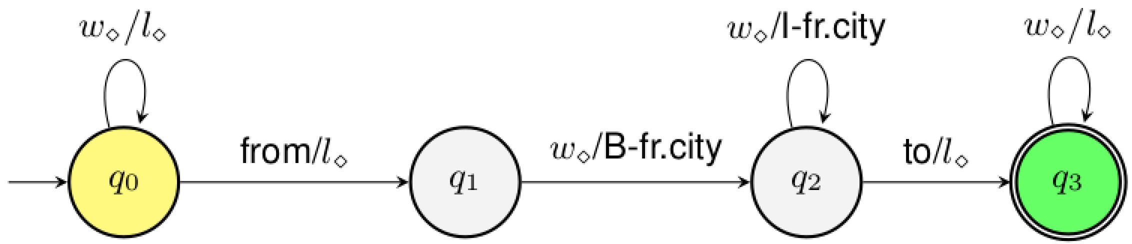
$$\boldsymbol{\beta}_{t-1} = \boldsymbol{\beta}_t \cdot \boldsymbol{T}'_\Omega[x_t]^T \quad \Rightarrow \quad \boldsymbol{\beta}_{t-1} = (\boldsymbol{\beta}_t \circ \boldsymbol{o}^T) \cdot \boldsymbol{T}[x_t]^T$$

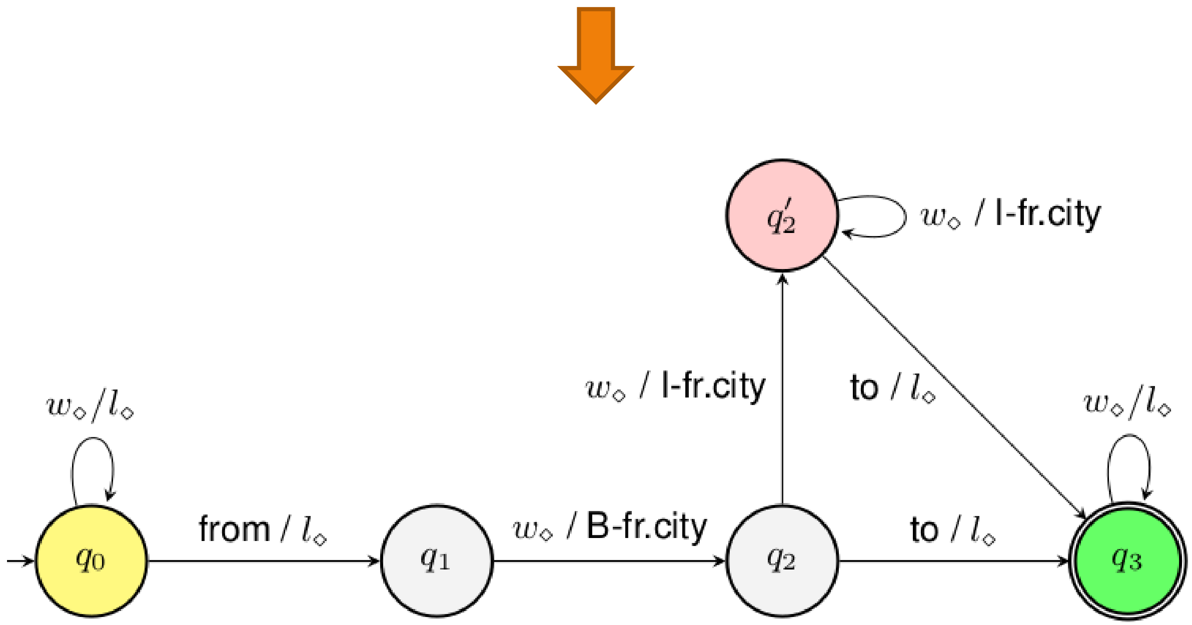
3. Label scoring

$$(\boldsymbol{c}_t)_k = (\boldsymbol{\alpha}_{t-1})_i \, (\boldsymbol{T}_\Omega[x_t])_{kij} \, (\boldsymbol{\beta}_t)_j \quad \Rightarrow \quad \boldsymbol{c}_t = (\boldsymbol{\alpha}_t \circ \boldsymbol{\beta}_t) \cdot \boldsymbol{O}$$

Time complexity per position: $O(LK^2) \rightarrow O(LK + 2K^2)$

Step 3. FST → iFST





Step 4&5. Tensor Decomposition & Pretrained Word Embedding

- Exactly the same as in the FA-RNN paper

$$\boldsymbol{\alpha}_t = (\boldsymbol{\alpha}_{t-1} \cdot \boxed{\boldsymbol{T}[x_t]}) \circ \boldsymbol{o}^T .$$

Rank Decomposition

$$\boxed{\boldsymbol{E}_{\mathcal{R}} \in \mathbb{R}^{V \times R}, \boldsymbol{D}_S \in \mathbb{R}^{K \times R}, \boldsymbol{D}_E \in \mathbb{R}^{K \times R}}$$

$$\boxed{\boldsymbol{v}_t = \boldsymbol{E}_{\mathcal{R}}[x_t]} \longrightarrow \boxed{\boldsymbol{v}_t = \eta \boldsymbol{E}_{\mathcal{R}}[x_t] + (1-\eta)\boldsymbol{E}_w[x_t] \cdot \boldsymbol{G}}$$

$$\boldsymbol{g} = (\boldsymbol{\alpha}_{t-1} \cdot \boldsymbol{D}_S) \circ \boldsymbol{v}_t$$

$$\boldsymbol{\alpha}_t = (\boldsymbol{g} \cdot \boldsymbol{D}_E^T) \circ \boldsymbol{o}^T$$

external word embedding

FST-RNN

FSTRNN Extensions

- Nonlinearity
- Dummy states
- Gating
  - Initialized to make the gates inactive initially
- Label priority
- CRF layer
  - Initialized with uniform transitions

Experiments

- Three slot-filling datasets
  - ATIS, ATIS-ZH, SNIPS
- Baselines:
  - Bi-RNN/GRU/LSTM
  - RE-enhanced NN (+i, +o, +io) [Luo et al., 2016]
  - Knowledge Distillation (+kd, +pr) [Hinton et al,.2015; Hu et al,. 2016]

Experiments – Zero-Shot

	Model	ATIS	ATIS-ZH	SNIPS
Softmax	*FSTRNN*	**73.10**	74.87	**52.02**
	FSTGRU	**73.10**	74.87	**52.02**
	BiGRU+(kd/pr/none)	1.76	1.91	0.69
	BiGRU+i	0.17	0.29	0.48
	BiGRU+o	11.70	22.88	10.49
CRF	*FSTRNN*	**73.10**	74.87	**52.02**
	FSTGRU	**73.10**	74.87	**52.02**
/	*RE*	72.36	**75.21**	51.98

Experiments – Low-Resource and Full Training

	Model	ATIS				ATIS-ZH				SNIPS				Average over Datasets			
		10	50	10%	100%	10	50	10%	100%	10	50	10%	100%	10	50	10%	100%
Softmax	***FSTRNN***	**74.59**	**74.94**	**85.43**	93.82	**75.09**	**75.25**	**82.25**	89.32	**51.94**	**52.84**	78.14	**90.15**	**67.21**	67.68	**81.94**	**91.10**
	BiRNN	57.11	65.80	80.93	**94.30**	63.56	65.07	81.90	**89.89**	17.03	39.37	77.58	87.62	45.90	56.75	80.14	90.60
	BiRNN+i	59.24	69.29	82.25	93.80	65.72	70.84	81.64	89.64	21.60	43.68	**79.45**	88.47	48.86	61.27	81.11	90.64
	BiRNN+o	54.64	66.44	80.63	93.91	64.89	65.79	81.29	89.24	16.87	39.75	76.75	87.95	45.47	57.33	79.56	90.37
	BiRNN+kd	54.21	65.45	81.44	94.18	63.31	65.10	81.80	89.77	17.60	39.56	78.47	88.83	45.04	56.70	80.57	90.93
	BiRNN+pr	55.21	68.28	81.13	93.91	63.56	65.07	81.90	89.73	17.85	39.56	77.50	88.13	45.54	57.64	80.18	90.59
	FSTGRU	**74.59**	**74.94**	**86.89**	94.74	**75.85**	**76.19**	**82.80**	90.50	**52.05**	**52.75**	80.50	90.92	**67.50**	**67.96**	**83.40**	92.05
	BiGRU	52.80	67.69	81.25	94.98	63.62	67.16	81.25	90.28	16.22	41.17	80.51	90.85	44.22	58.67	81.00	92.03
	BiGRU+i	57.68	69.87	83.11	94.63	*64.55*	71.96	82.02	90.16	20.33	44.12	**81.17**	90.70	47.52	61.99	82.10	91.83
	BiGRU+o	52.67	66.97	80.73	94.93	63.29	68.54	80.90	90.13	16.84	40.56	80.44	91.05	44.27	58.69	80.69	92.03
	BiGRU+kd	53.49	67.23	80.99	**95.04**	63.90	67.23	81.36	**90.70**	17.85	41.44	80.16	**91.40**	45.08	58.63	80.84	**92.38**
	BiGRU+pr	52.77	67.69	81.25	94.98	63.35	67.16	81.25	90.28	17.49	41.30	79.94	91.19	44.53	58.72	80.81	92.15
CRF	***FSTRNN***	**74.61**	**74.76**	**85.94**	94.09	**76.08**	**75.92**	**82.92**	90.07	**51.77**	**52.83**	80.77	91.78	**67.49**	**67.83**	**83.21**	**91.98**
	BiRNN	55.04	70.75	82.06	94.30	62.96	67.04	82.82	89.93	17.19	40.47	80.21	90.21	45.07	59.42	81.70	91.48
	BiRNN+i	58.25	69.37	83.84	94.02	65.75	71.40	82.68	89.59	22.18	45.26	**81.90**	**92.24**	48.73	62.01	82.81	91.95
	BiRNN+o	55.26	67.88	83.77	**94.32**	61.47	67.21	82.42	**90.13**	16.81	40.63	79.96	90.45	44.51	58.58	82.05	91.64
	BiRNN+kd	54.93	69.08	82.46	93.58	62.92	67.04	82.84	89.71	17.34	40.48	80.31	90.33	45.06	58.87	81.87	91.21
	BiRNN+pr	56.16	68.02	82.77	93.58	62.96	67.04	82.84	89.64	17.30	40.23	80.47	90.74	45.47	58.43	82.03	91.32
	FSTGRU	**74.61**	**74.76**	**86.50**	95.00	**75.85**	**75.92**	**83.48**	90.73	**52.05**	**53.01**	81.98	**93.17**	**67.50**	**67.89**	**83.99**	**92.97**
	BiGRU	54.43	67.22	79.11	94.66	64.27	68.72	82.71	90.55	18.13	42.47	82.88	92.77	45.61	59.47	81.57	92.66
	BiGRU+i	57.57	70.67	84.44	94.72	64.20	71.43	83.39	90.45	20.76	46.34	**83.30**	92.94	47.51	62.81	83.71	92.70
	BiGRU+o	54.40	67.39	83.24	95.02	63.12	69.27	82.49	90.48	17.40	41.64	82.82	92.49	44.97	59.43	82.85	92.66
	BiGRU+kd	53.31	68.14	82.17	95.22	62.12	68.72	82.52	90.52	17.06	42.47	83.38	92.70	44.17	59.78	82.69	92.81
	BiGRU+pr	53.41	68.34	82.15	**95.39**	62.46	68.72	82.71	**90.75**	17.01	42.47	83.21	92.75	44.29	59.84	82.69	92.96

Part 1 Summary

- FA-RNN / FST-RNN combines strengths of symbolic rules and neural networks
  - Can be converted from RE
  - Can also learn from labeled data
  - Excels in zero-shot and low-resource scenarios; competitive in rich-resource scenarios

Outline

- Symbolism vs. Connectionism
- Turning symbolic systems to neural networks
- Learning symbolic systems using neural networks

Learning symbolic systems using neural networks

- Goal: learning symbolic rules from scratch
- Running example: grammar induction
  - Grammar: a set of rules (with probabilities)
  - Induction: unsupervised learning

- Outline
  - Introduction of grammar induction
  - Unfold inference as neural networks
  - Symbol embedding and neural parameterization
  - Contextualize grammar rules

Context-Free Grammars

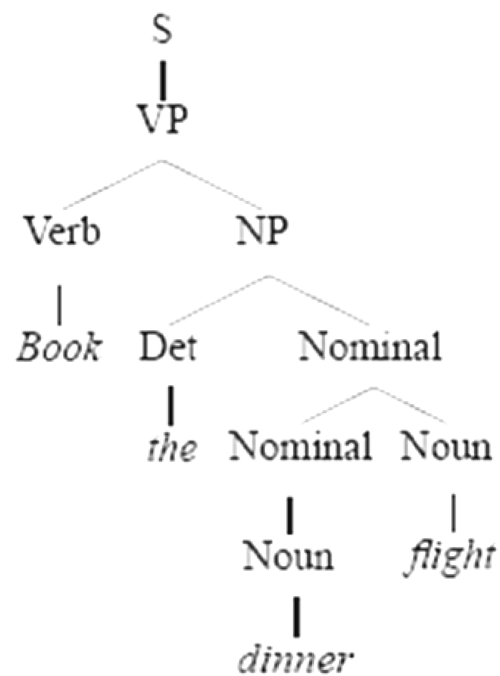
- A context-free grammar (CFG) has four components
  - A set Σ of terminals (words)
  - A set N of nonterminals (phrases)
  - A start symbol $S \in N$
  - A set R of production rules
    - Specifies how a nonterminal can produce a string of terminals and/or nonterminals

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
S &\rightarrow Aux\ NP\ VP \\
S &\rightarrow VP \\
NP &\rightarrow Pronoun \\
NP &\rightarrow Proper\text{-}Noun \\
NP &\rightarrow Det\ Nominal \\
NP &\rightarrow Nominal \\
Nominal &\rightarrow Noun \\
Nominal &\rightarrow Nominal\ Noun \\
Nominal &\rightarrow Nominal\ PP \\
VP &\rightarrow Verb \\
VP &\rightarrow Verb\ NP \\
VP &\rightarrow Verb\ NP\ PP \\
VP &\rightarrow Verb\ PP \\
VP &\rightarrow Verb\ NP\ NP \\
VP &\rightarrow VP\ PP \\
PP &\rightarrow Preposition\ NP
\end{aligned}
$$

Generation & Parsing

$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$NP \rightarrow Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow Verb\ NP\ NP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

……



Book the dinner flight

Example from [Jurafsky & Martin, 2006]

Probabilistic Grammars

- Each rule is associated with a conditional probability

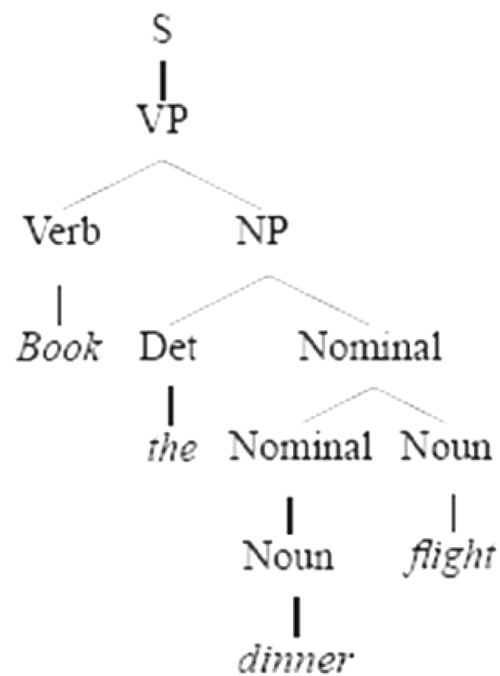
$$\alpha \rightarrow \beta : P(\alpha \rightarrow \beta | \alpha)$$

- The probability of a parse tree is the product of the probabilities of all the rules used in generating the parse tree

Example

$S \rightarrow NP\ VP$		[.80]
$S \rightarrow Aux\ NP\ VP$		[.15]
$S \rightarrow VP$		[.05]
$NP \rightarrow Pronoun$		[.35]
$NP \rightarrow Proper\text{-}Noun$		[.30]
$NP \rightarrow Det\ Nominal$		[.20]
$NP \rightarrow Nominal$		[.15]
$Nominal \rightarrow Noun$		[.75]
$Nominal \rightarrow Nominal\ Noun$		[.20]
$Nominal \rightarrow Nominal\ PP$		[.05]
$VP \rightarrow Verb$		[.35]
$VP \rightarrow Verb\ NP$		[.20]
$VP \rightarrow Verb\ NP\ PP$		[.10]
$VP \rightarrow Verb\ PP$		[.15]
$VP \rightarrow Verb\ NP\ NP$		[.05]
$VP \rightarrow VP\ PP$		[.15]
$PP \rightarrow Preposition\ NP$		[1.0]

……



Book the dinner flight

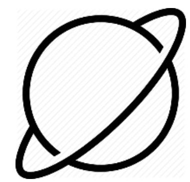
$$P(T) = .05 \times .20 \times .20 \times .20 \times .75 \times .30 \times .60 \times .10 \times .40 = 2.2 \times 10^{-6}$$

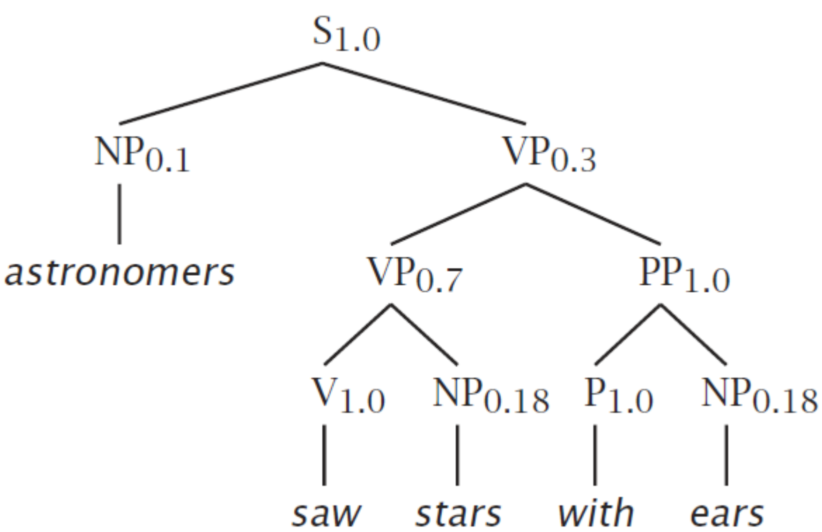
Parse tree scoring

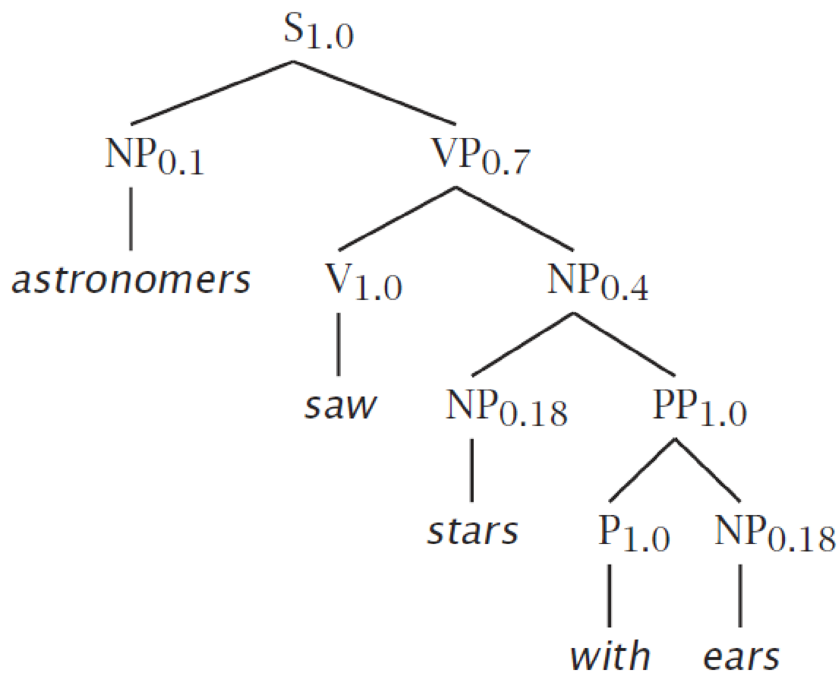
- Assign a probability (or score) to a parse tree of a sentence
- Why?
  - Disambiguation!
  - A natural language sentence may have many possible parses
  - Ambiguities are ubiquitous in natural languages

Ambiguity

- Astronomers saw stars with ears.



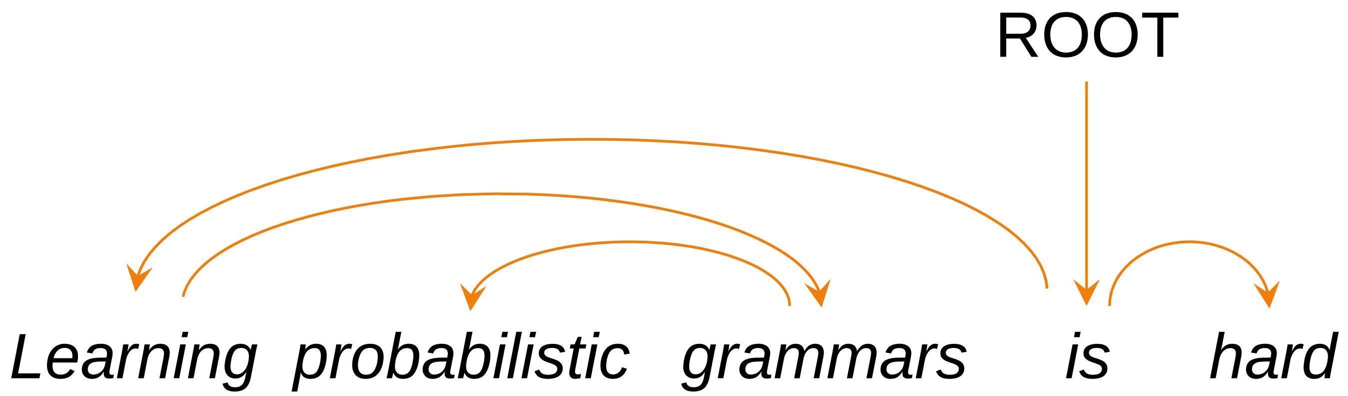




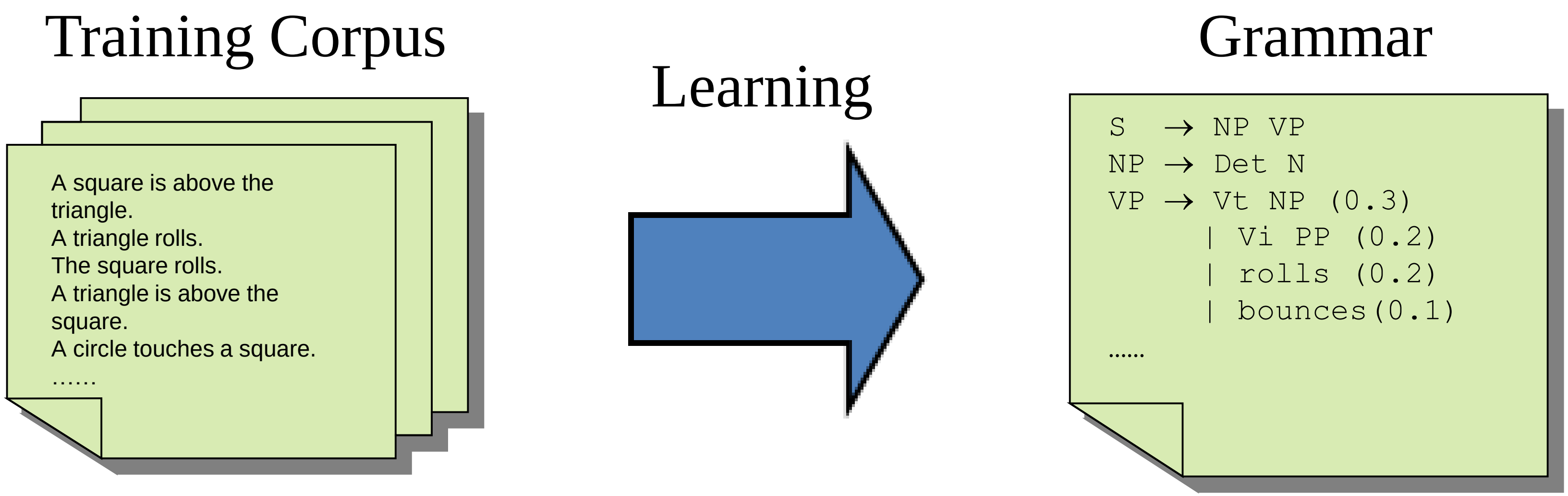
Example from [Manning and Schütze, 1999]

Dependency Grammar

- Dependency grammar & parsing
  - ROOT → is,   ROOT → give,   …
  - is+left → learning,   is+right → hard,   …



Learning a grammar from a corpus



- Supervised Methods
  - Rely on a training corpus of sentences annotated with parses (treebank)
- Unsupervised Methods (Grammar Induction)
  - Do not require annotated data

Grammar Induction

- Learn a grammar from *unannotated* sentences
- Two subtasks
  - Structure search
    - Learn a set of grammar rules
  - Parameter learning
    - Given a set of grammar rules, learn their probabilities

Extremely difficult on real data. Almost no success.

Still difficult, but doable. A lot of work over the past 20yrs.

Learning symbolic systems using neural networks

- Outline
  - Introduction of grammar induction
  - Unfold inference as neural networks
    - Songlin Yang, Yanpeng Zhao, and Kewei Tu, "PCFGs Can Do Better: Inducing Probabilistic Context-Free Grammars with Many Symbols", NAACL 2021.
    - Songlin Yang, Yanpeng Zhao, and Kewei Tu, "Neural Bi-Lexicalized PCFG Induction", ACL 2021.
  - Symbol embedding and neural parameterization
  - Contextualize grammar rules

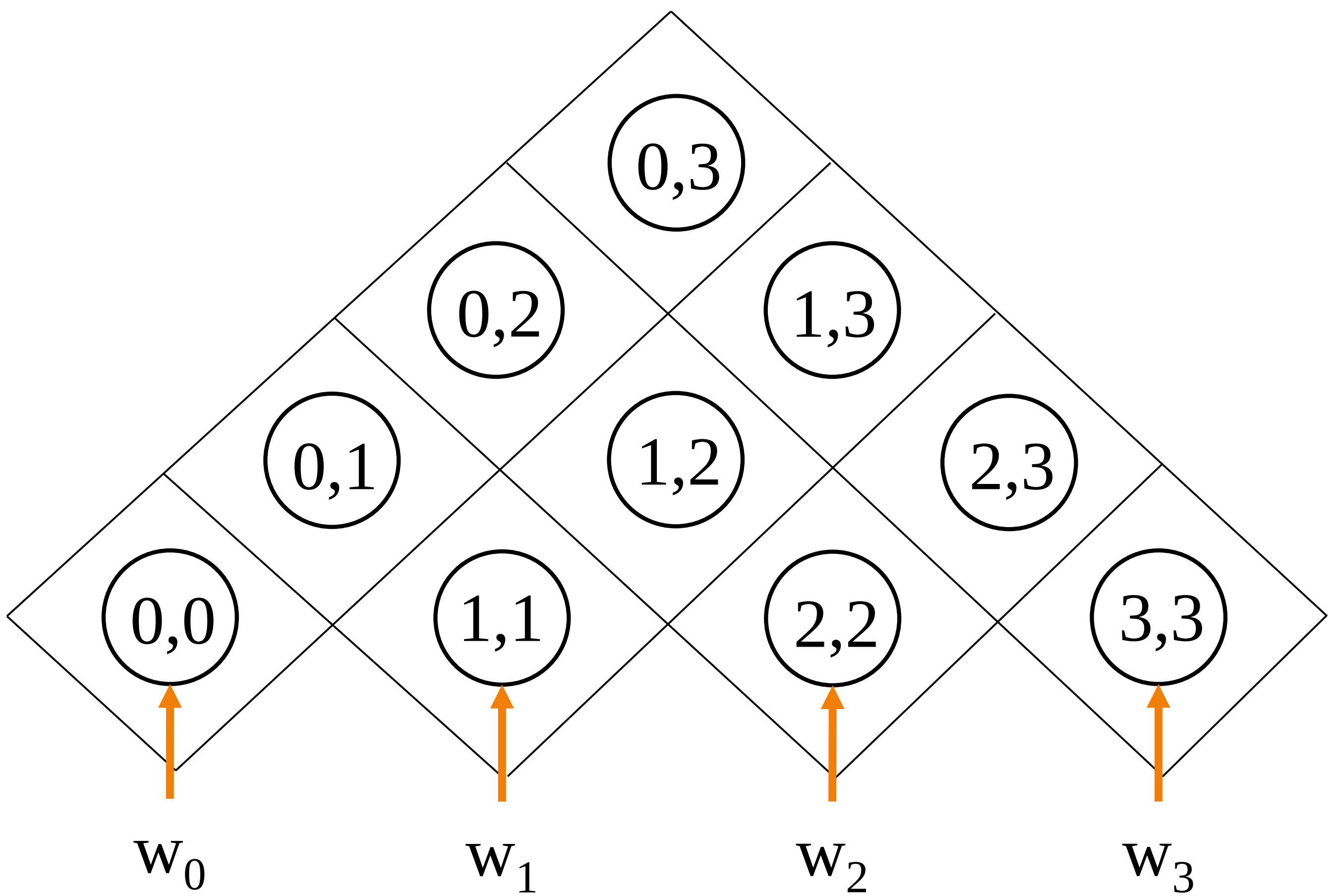
Parameter Learning

- Typical objective function: MLE

$$F(\theta) = \log P_\theta(\boldsymbol{w}) = \log \sum_{t \in T(\boldsymbol{w})} P_\theta(t) = \log \sum_{t \in T(\boldsymbol{w})} \prod_{r \in t} \theta_r$$

  - Can be computed with dynamic programming (the inside algorithm)
  - Traditionally optimized using the EM algorithm
    - Non-trivial to understand, implement, and parallelize
  - Optimization with gradient descent?

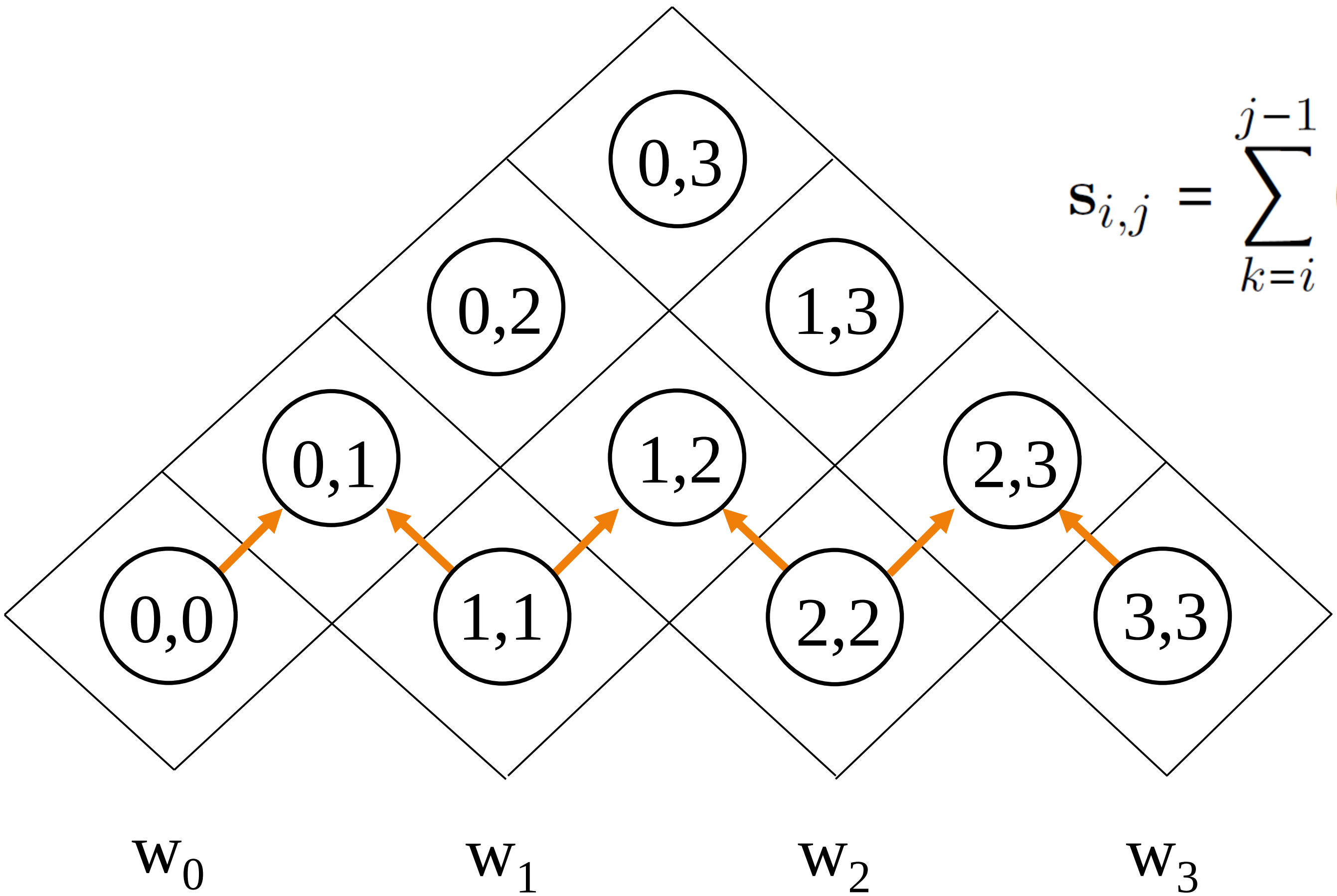
Computation graph of the inside algorithm



Probabilities of preterminal rules (A→w)

$$\mathbf{s}_{i,i} = \mathbf{Q}(w_i)$$

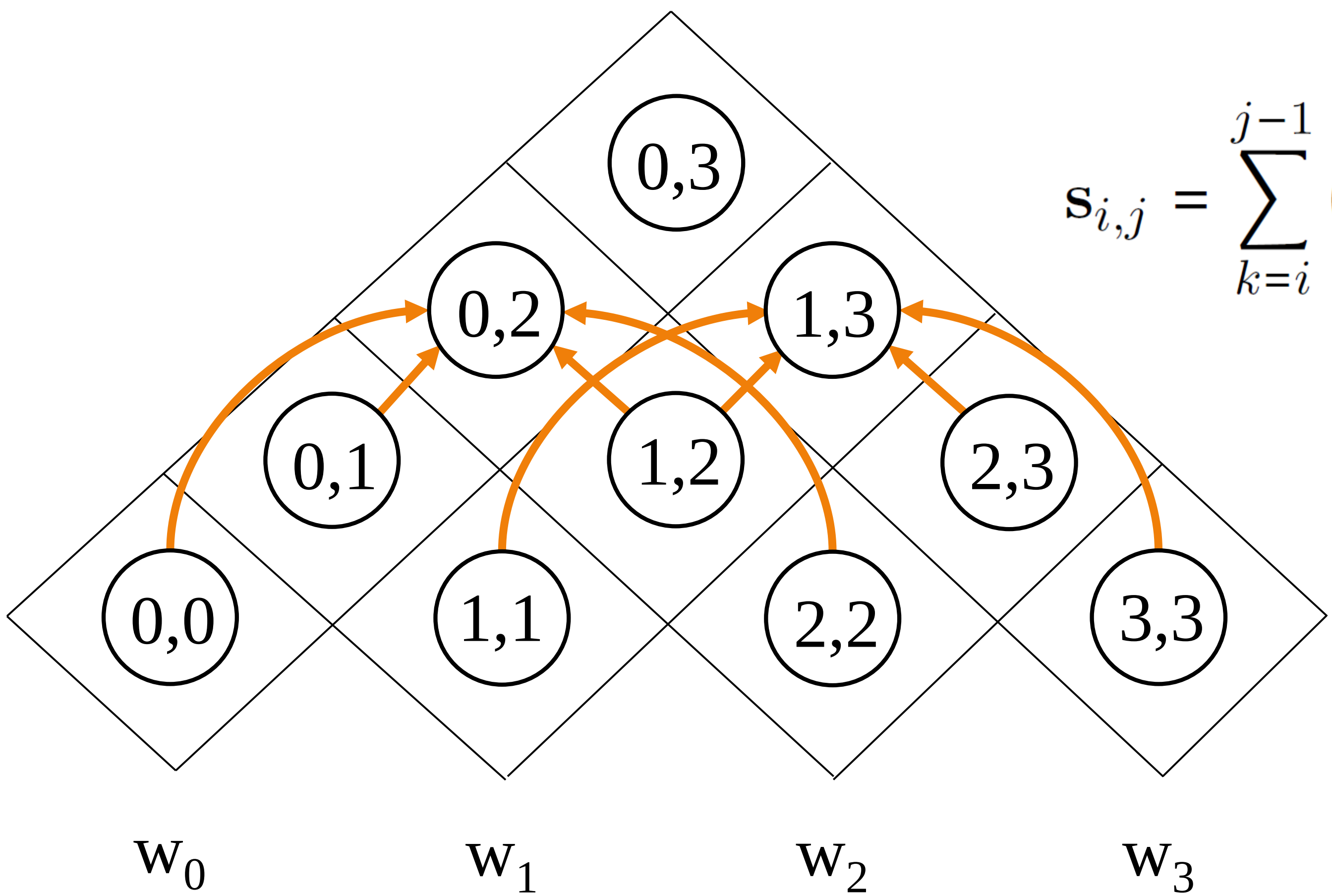
Computation graph of the inside algorithm



$$\mathbf{s}_{i,j} = \sum_{k=i}^{j-1} \left(\mathbf{T} \cdot \mathbf{s}_{k+1,j} \right) \cdot \mathbf{s}_{i,k}$$

Probabilities of binary rule (A→BC)

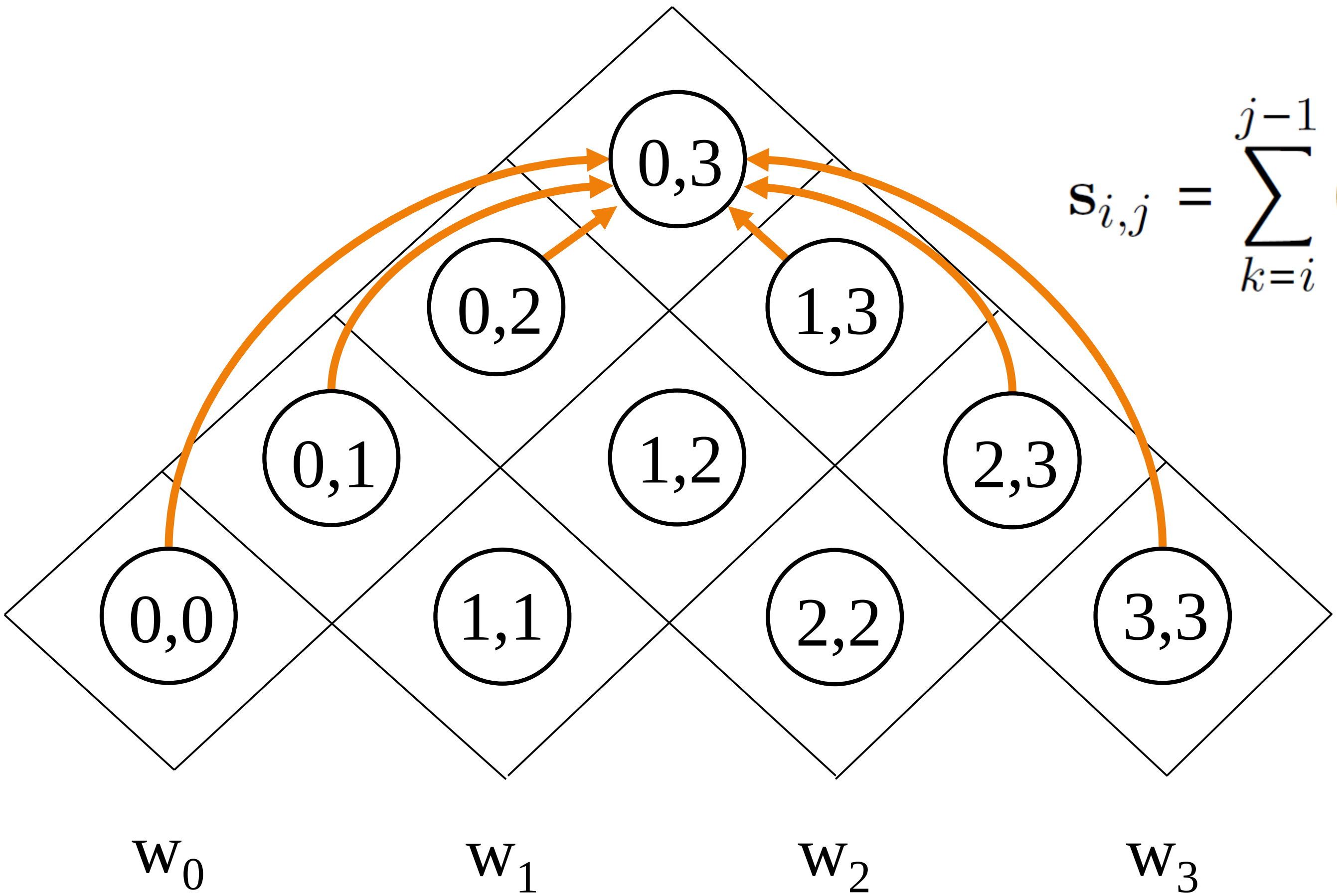
Computation graph of the inside algorithm



$$\mathbf{s}_{i,j} = \sum_{k=i}^{j-1} \left(\mathbf{T} \cdot \mathbf{s}_{k+1,j}\right) \cdot \mathbf{s}_{i,k}$$

Probabilities of binary rule (A→BC)

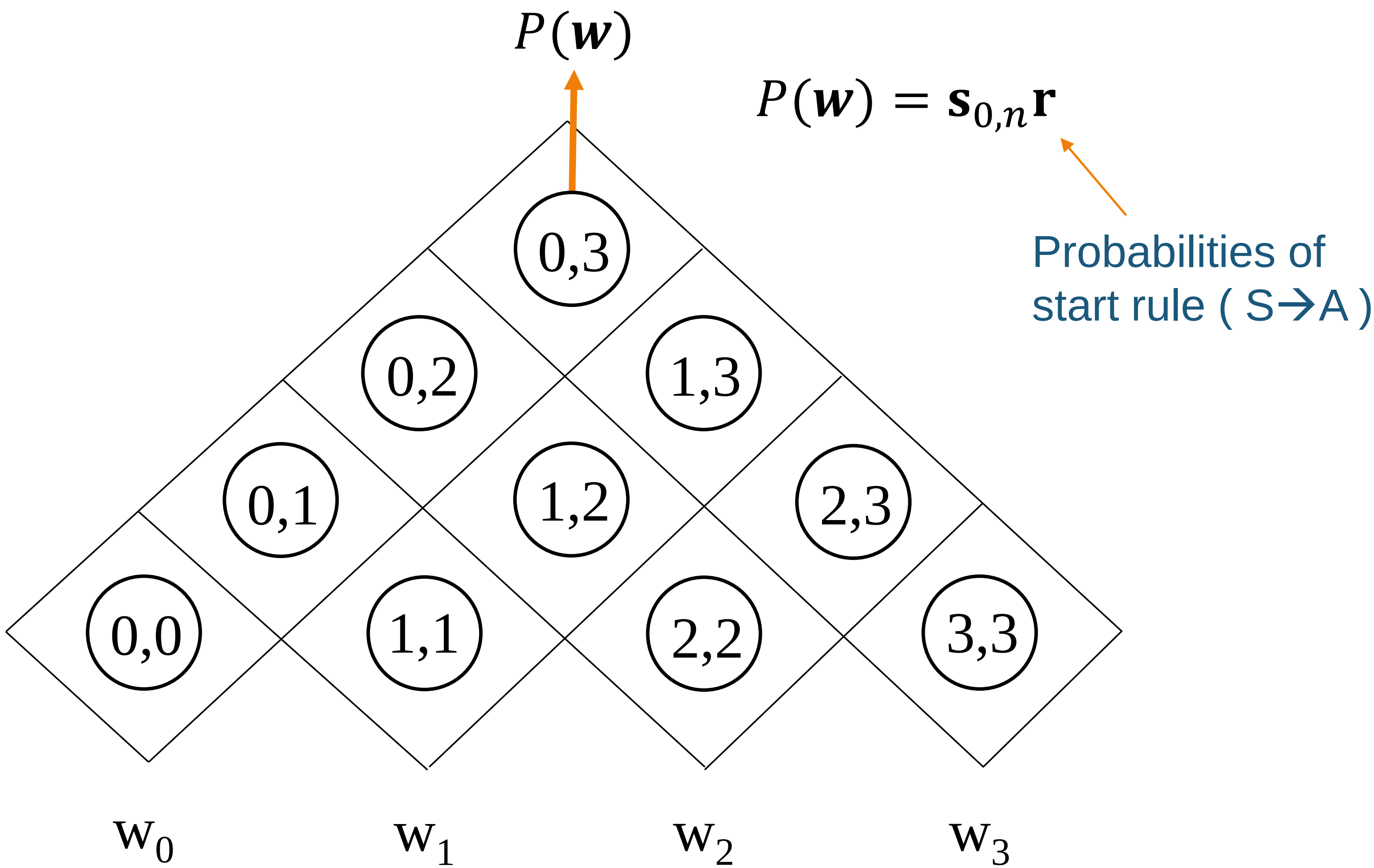
Computation graph of the inside algorithm



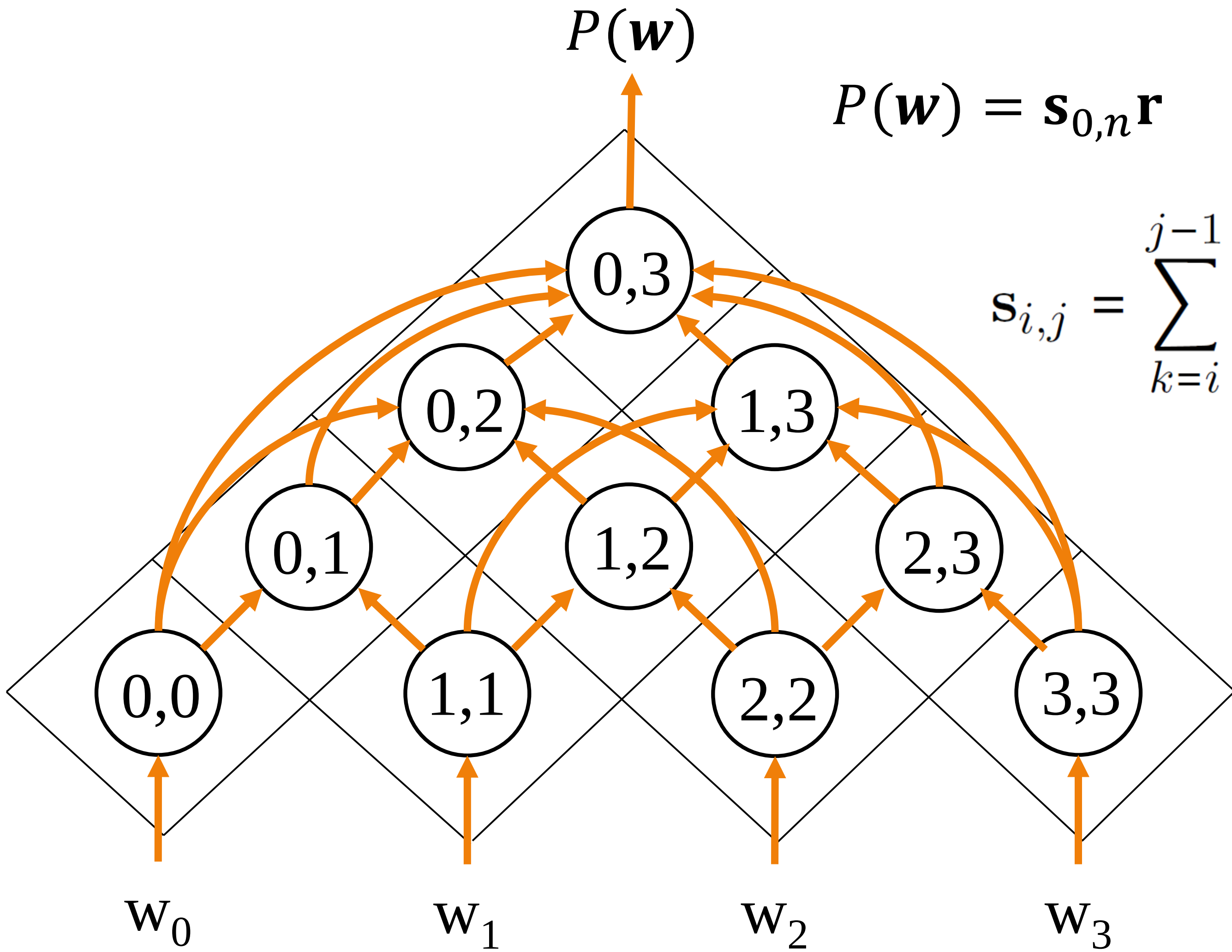
$$\mathbf{s}_{i,j} = \sum_{k=i}^{j-1} \left(\mathbf{T} \cdot \mathbf{s}_{k+1,j}\right) \cdot \mathbf{s}_{i,k}$$

Probabilities of binary rule (A→BC)

Computation graph of the inside algorithm



Computation graph of the inside algorithm



$O(m^3 l^3)$ time & space complexity!

$$P(\boldsymbol{w}) = \mathbf{s}_{0,n}\mathbf{r}$$

$$\mathbf{s}_{i,j} = \sum_{k=i}^{j-1} \left(\mathbf{T} \cdot \mathbf{s}_{k+1,j}\right) \cdot \mathbf{s}_{i,k}$$

$$\mathbf{s}_{i,i} = \mathbf{Q}(w_i)$$

Reducing Complexity

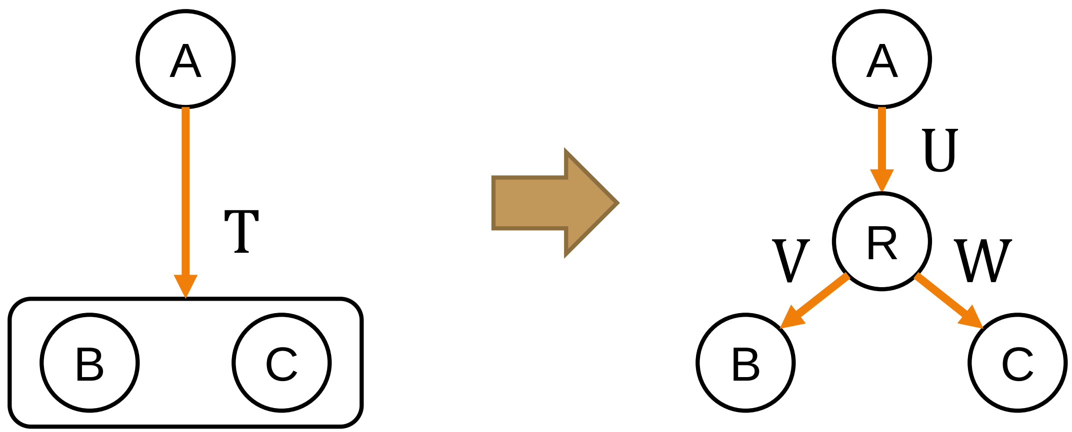
- Kruskal form of T:

$$\mathbf{T} = \sum_{l=1}^{d} \mathbf{T}^{(l)}, \quad \mathbf{T}^{(l)} = \mathbf{u}^{(l)} \otimes \mathbf{v}^{(l)} \otimes \mathbf{w}^{(l)}$$

- Neural parameterization of U, V, W
  - More on this later…
- Problem: T contains probabilities
- Solution:
  - V and W are column-normalized
  - U is row-normalized

A Bayesian network perspective

- A→BC



Reducing Complexity

- Simplified update formula:

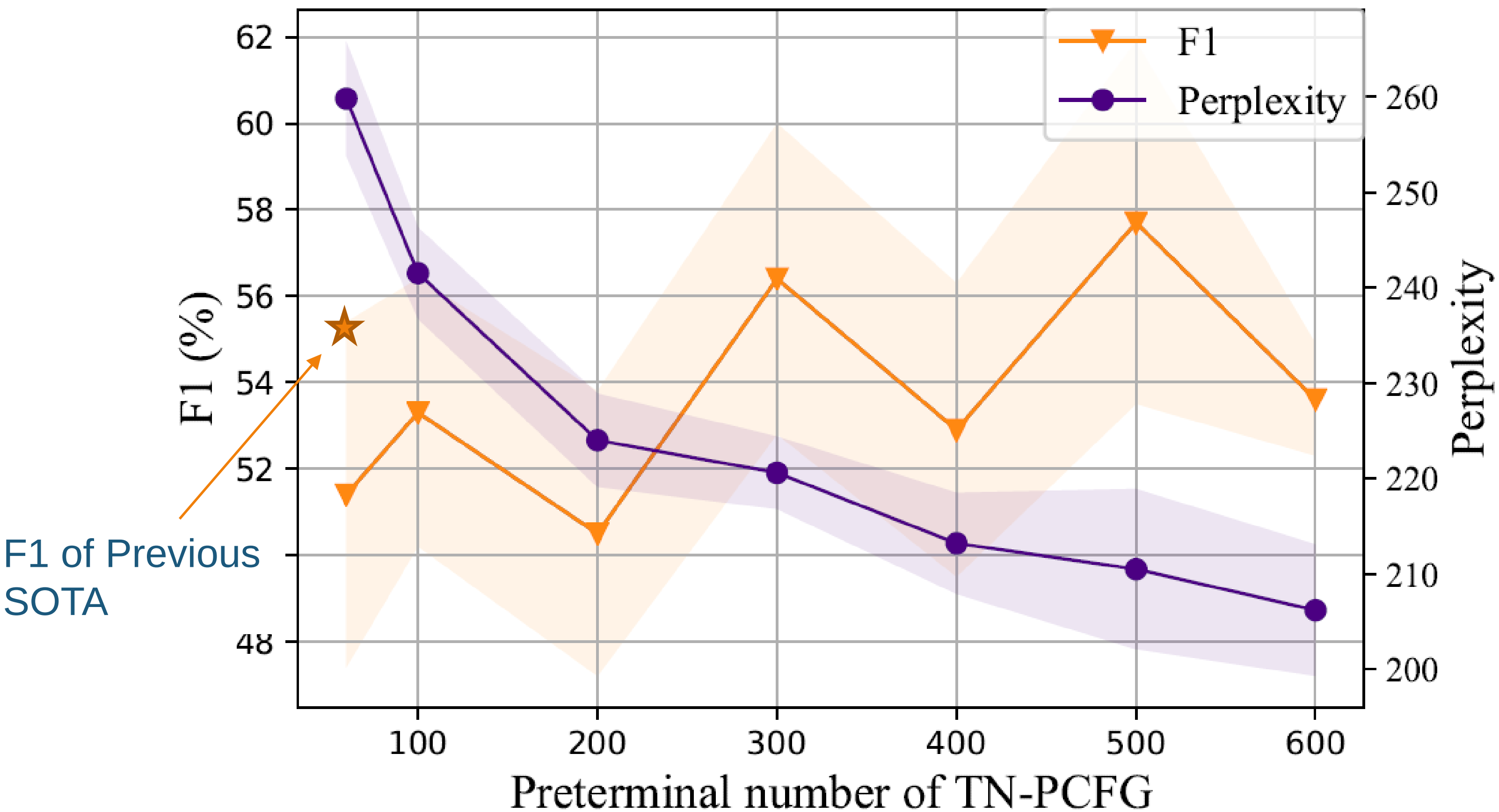
$$\mathbf{s}_{i,j} = \sum_{k=i}^{j-1} \left(\mathbf{T} \cdot \mathbf{s}_{k+1,j}\right) \cdot \mathbf{s}_{i,k}$$

$$\Downarrow$$

$$\mathbf{s}_{i,j} = \mathbf{U} \cdot \sum_{k=i}^{j-1} \left(\left(\mathbf{V}^T \mathbf{s}_{i,k}\right) \odot \left(\mathbf{W}^T \mathbf{s}_{k+1,j}\right)\right)$$

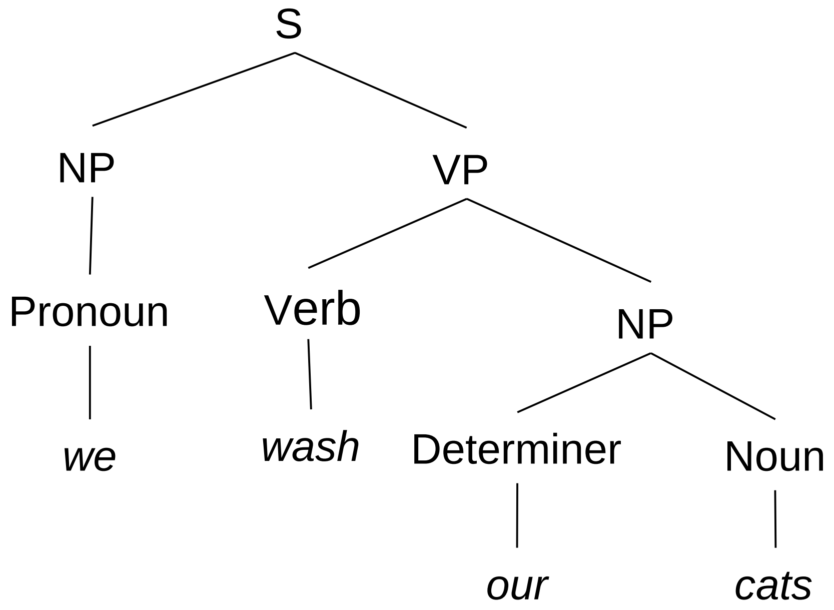
- Reduced complexity: $O(dl^3 + mdl^2)$
- A form of *recursive neural networks* if given a parse tree!

Experimental results



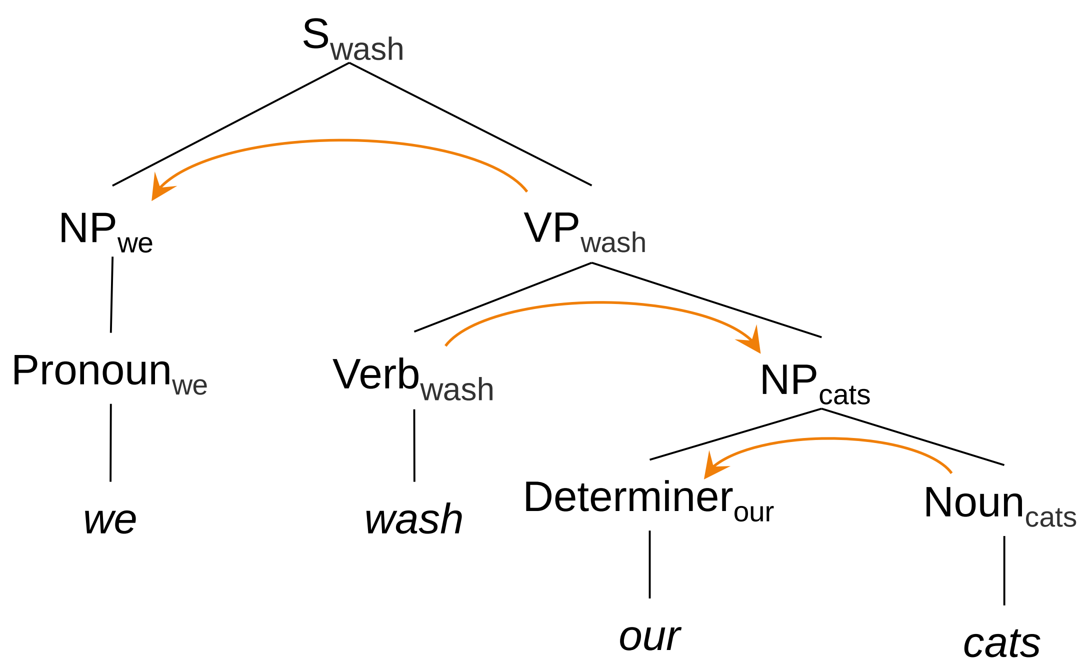
Extension to Bilexical PCFG

- Unlexicalized production rules:
  - A → B C



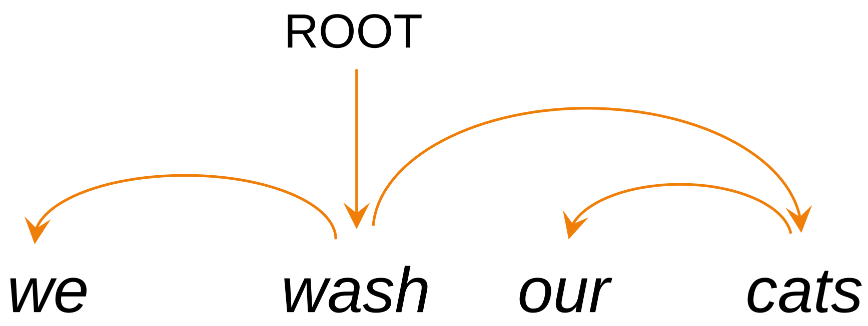
Extension to Bilexical PCFG

- Lexicalized production rules:
  - $A[w_p] \rightarrow B[w_p]\ C[w_q]$ or $A[w_p] \rightarrow B[w_q]\ C[w_p]$



Extension to Bilexical PCFG

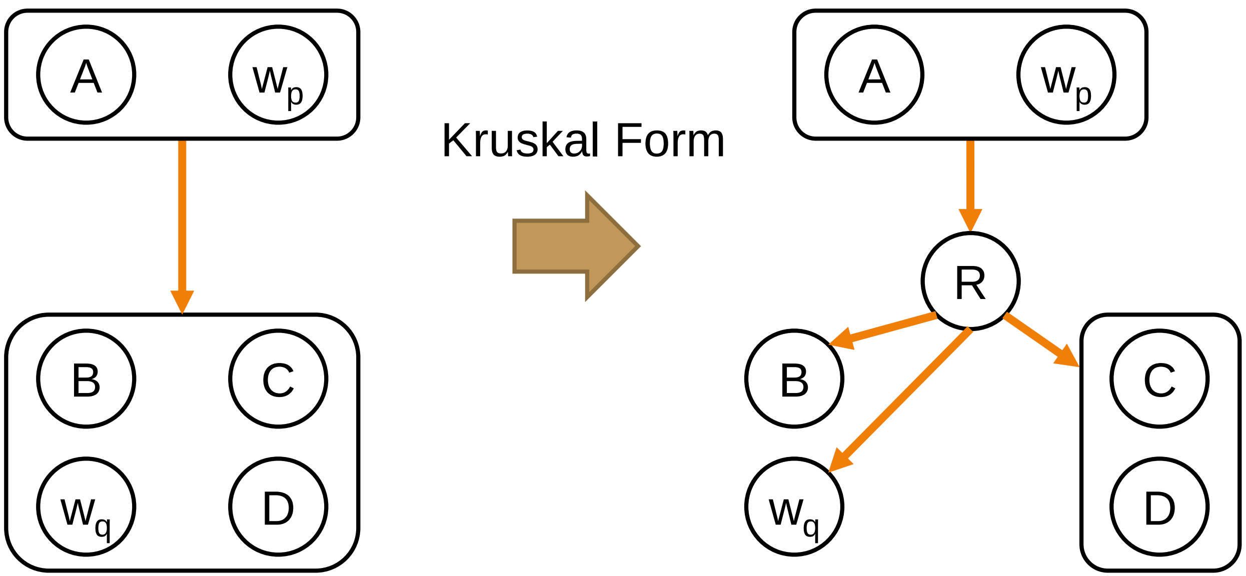
- Lexicalized production rules:
  - $A[w_p] \rightarrow B[w_p]\ C[w_q]$ or $A[w_p] \rightarrow B[w_q]\ C[w_p]$



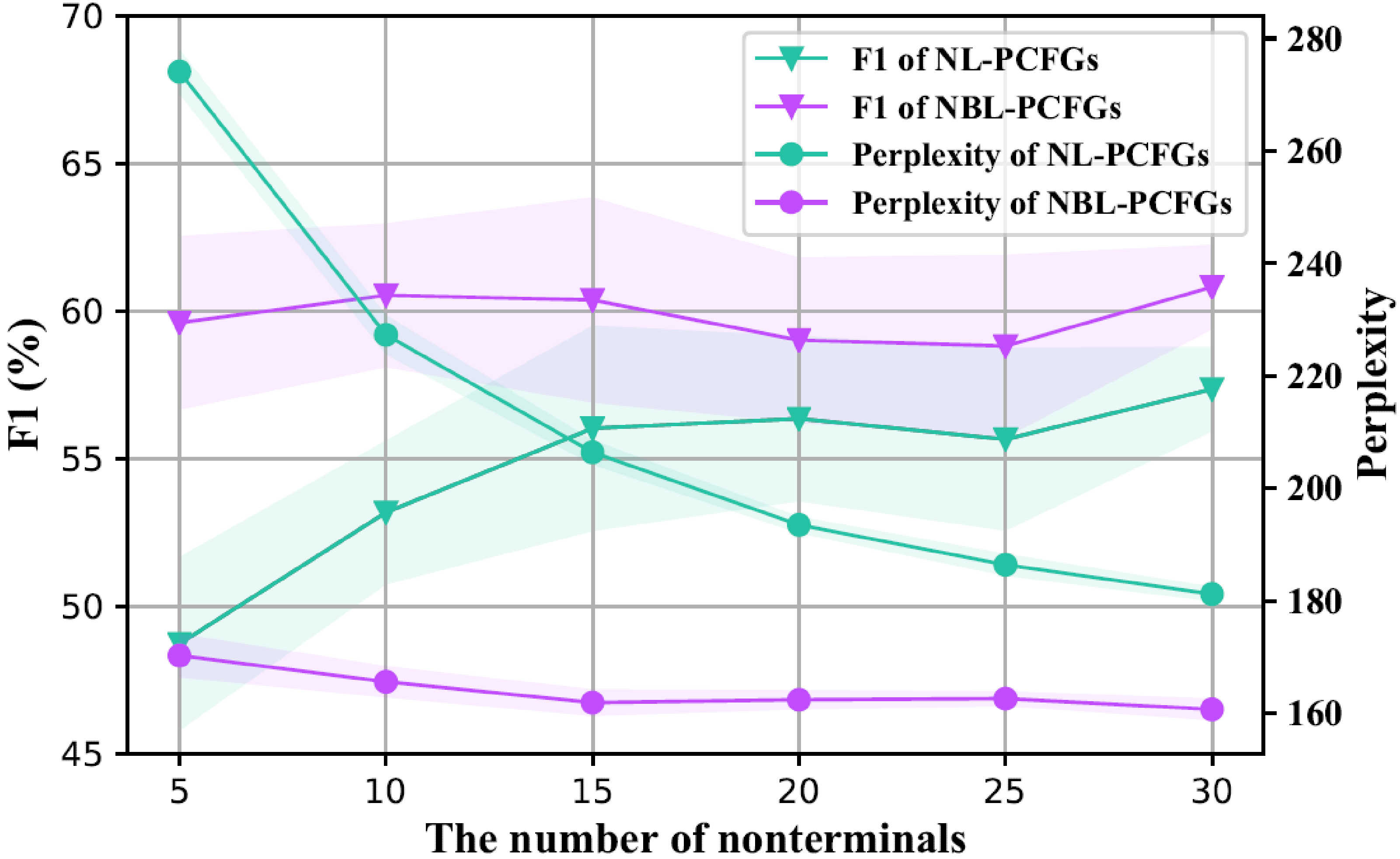
- A bilexical CFG can simultaneously produce a constituency parse and a dependency parse

Extension to Bilexical PCFG

- Lexicalized production rules:
  - $A[w_p] \rightarrow B[w_p]\ C[w_q]$ or $A[w_p] \rightarrow B[w_q]\ C[w_p]$
  - Rule probability:



Experimental results



Learning symbolic systems using neural networks

- Outline
  - Introduction of grammar induction
  - Unfold inference as neural networks
  - Symbol embedding and neural parameterization
    - Yong Jiang, Wenjuan Han, and Kewei Tu, "Unsupervised Neural Dependency Parsing", EMNLP 2016.
  - Contextualize grammar rules

Problem

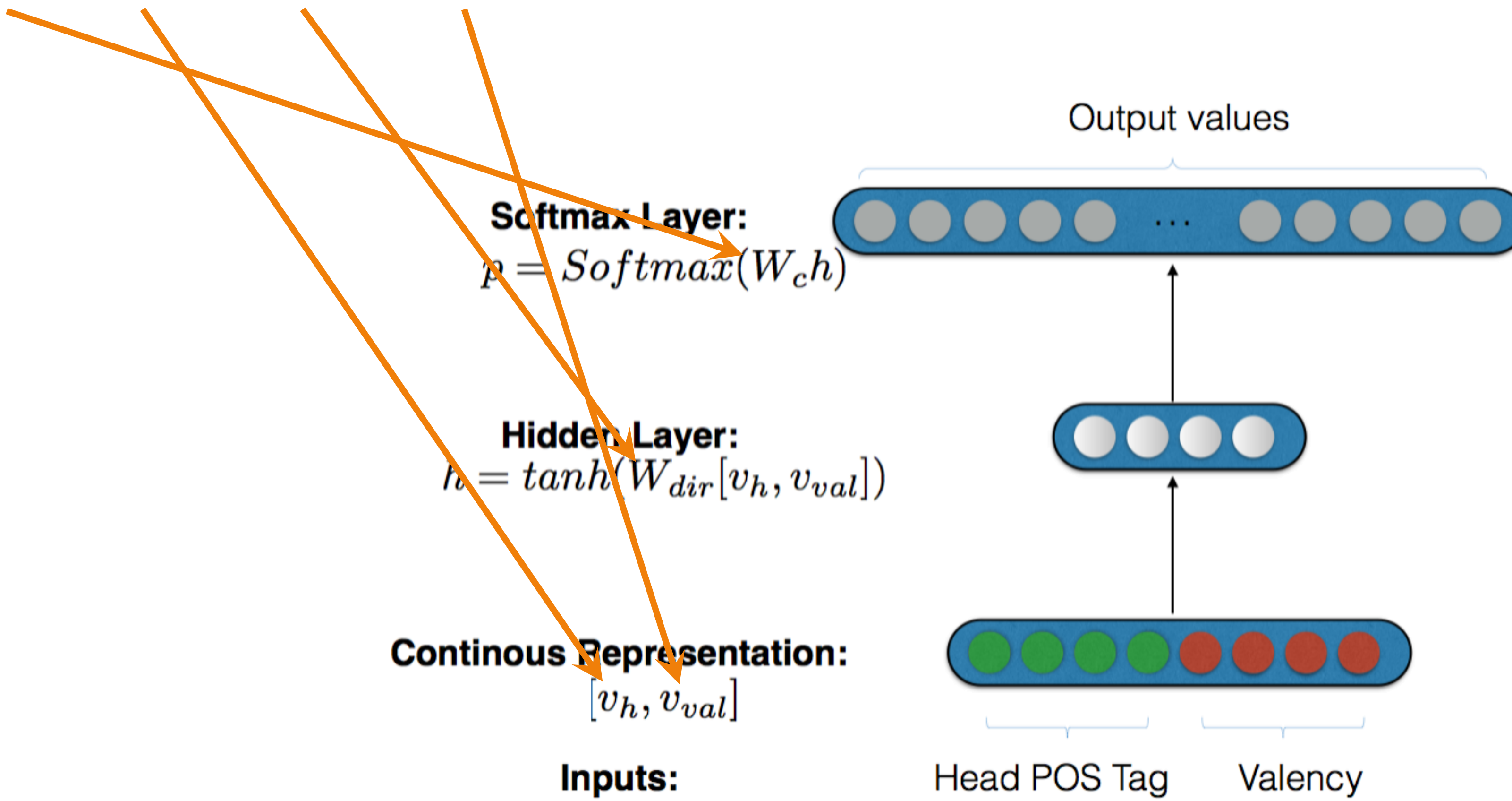
- Different terminal/nonterminal symbols in a grammar are regarded as being distinct
- But correlations exist between many of them
  - Example: verb base form, past tense, 3rd person singular (subtypes of the same parent type)

Problem

- Solution: symbol embedding
  - Learn to embed terminal/nonterminal symbols into a continuous vector space
  - Similar symbols are close to each other in the embedding space
  - Predict grammar rule weights or parsing actions from the vector representations of the grammar symbols

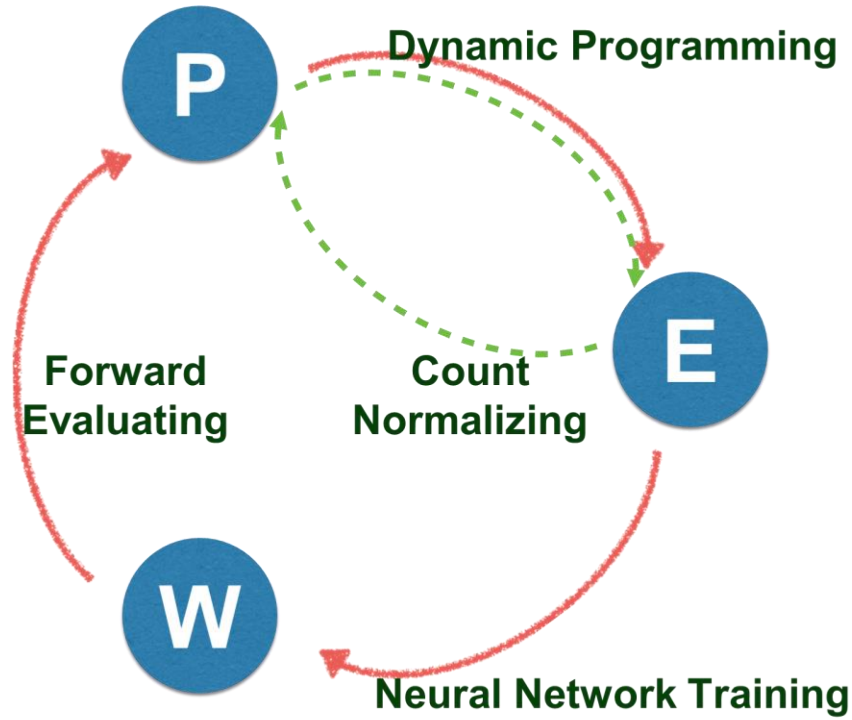
Neural DMV

$$P(child \mid head, direction, valency)$$



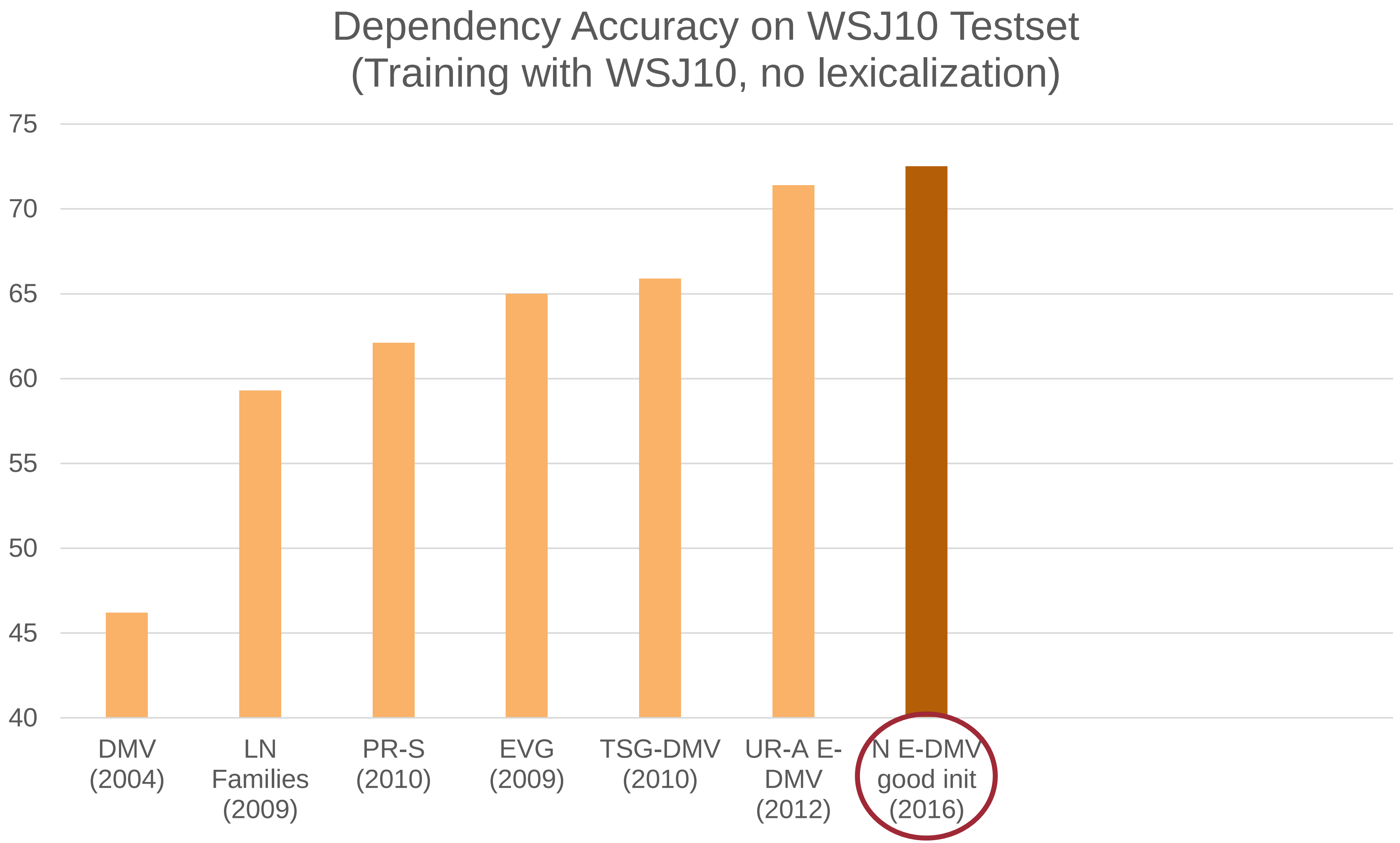
Learning Neural DMV

- Expectation-Maximization



- Gradient descent can be even better!
  - Songlin Yang, Yong Jiang, Wenjuan Han, and Kewei Tu, "Second-Order Unsupervised Neural Dependency Parsing", COLING 2020

Experimental results



Generative Approaches

500
400
300
200
100
0
-100
-200
-300
-400
-500
-500
-400
-300
-200
-100
0
100
200
300
400
500
Adj
Adv
Noun
Others
Verb
VBP
VBD
VBZ
MD
$
VBN
RBS
RBR
RP
WP$
WP
PDT
VBG
UH
WRB
PRP
T0
IN
EX
WDT
VB
FW
RB
POS
JJS
NNPS
JJ
NNP
JJR
NNS
CD
PRP$
NN
DT
CC

Learning symbolic systems using neural networks

- Outline
  - Introduction of grammar induction
  - Unfold inference as neural networks
  - Symbol embedding and neural parameterization
  - Contextualize grammar rules
    - Wenjuan Han, Yong Jiang, and Kewei Tu, "Enhancing Unsupervised Generative Dependency Parser with Contextual Information", ACL 2019.

Another Problem

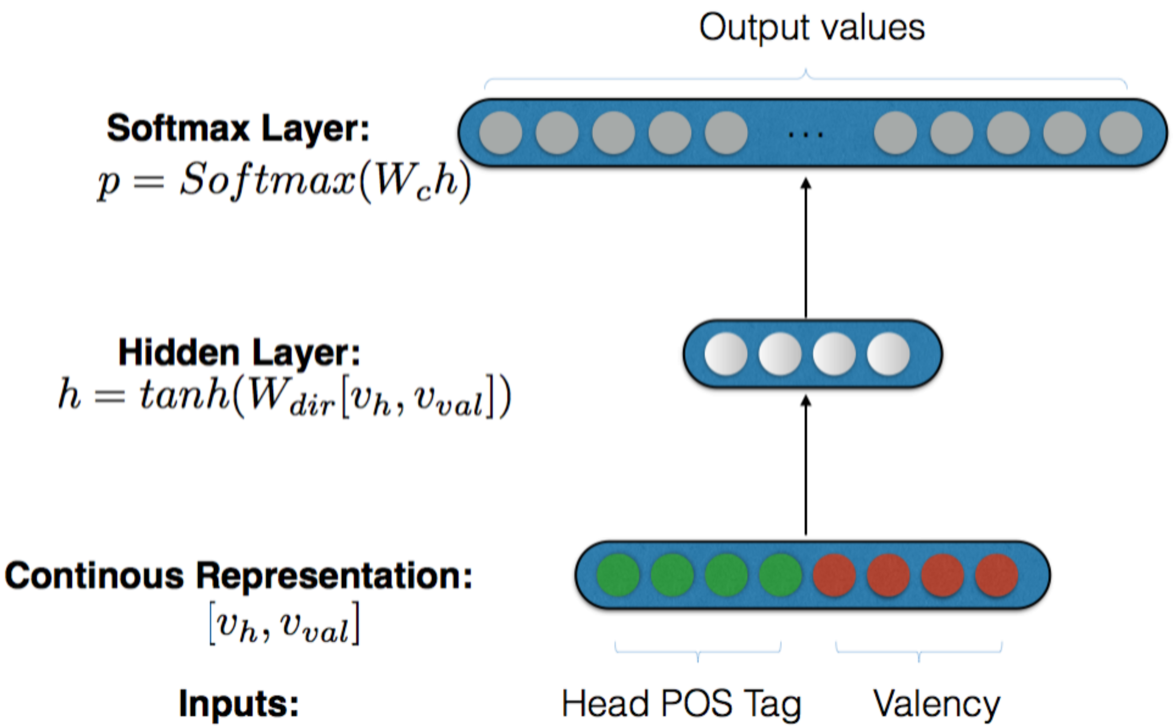
- The same grammar rule may have different probabilities in different contexts
  - “He is reading a book.” vs. “What is he reading?”

dobj dobj

Discriminative parsing

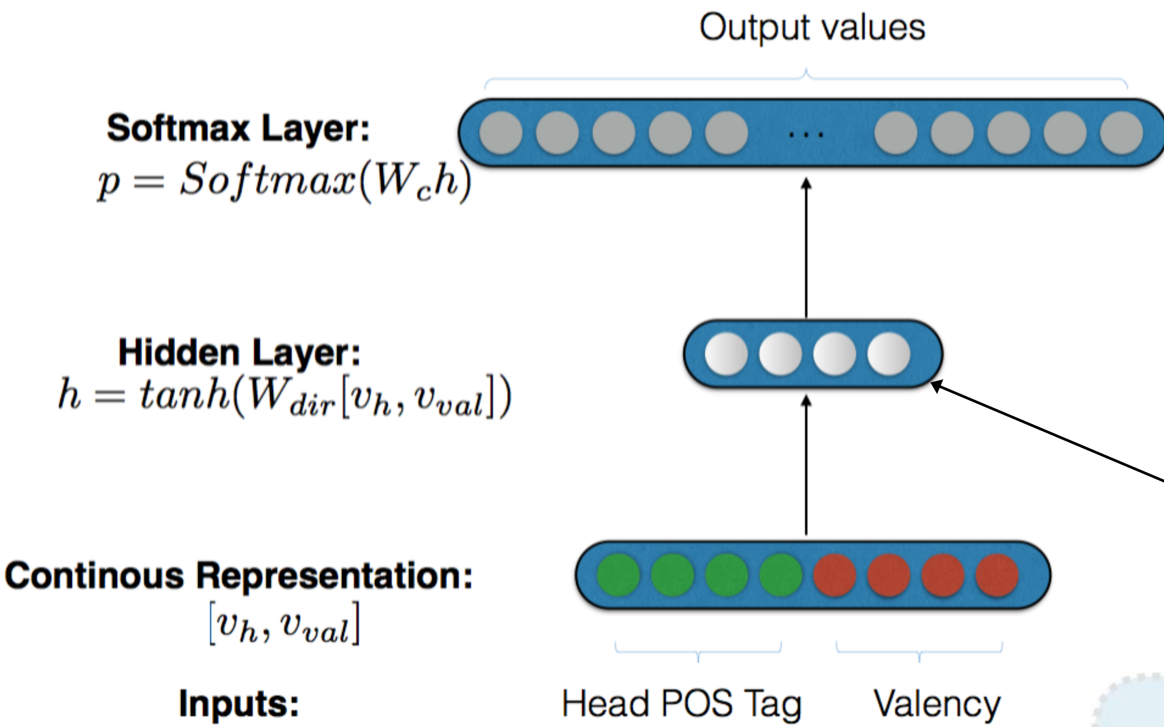
- A discriminative parser models P(parse | sentence)
  - Utilize rich features of the whole sentence in predicting the parse tree
  - Grammar rule probabilities or weights depend on the context

Neural DMV

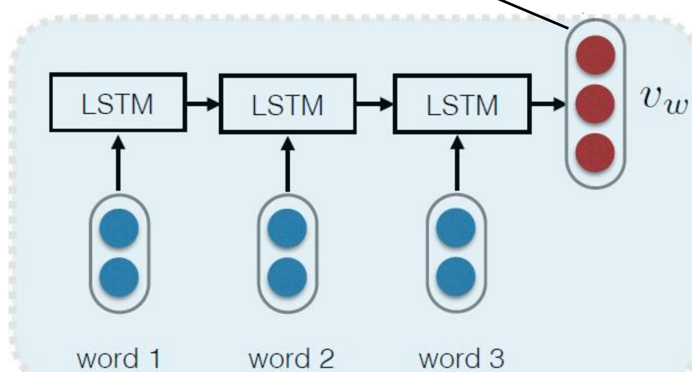


N-DMV objective: $J(\theta) = \sum_y P_\theta(x, y)$

Discriminative Neural DMV



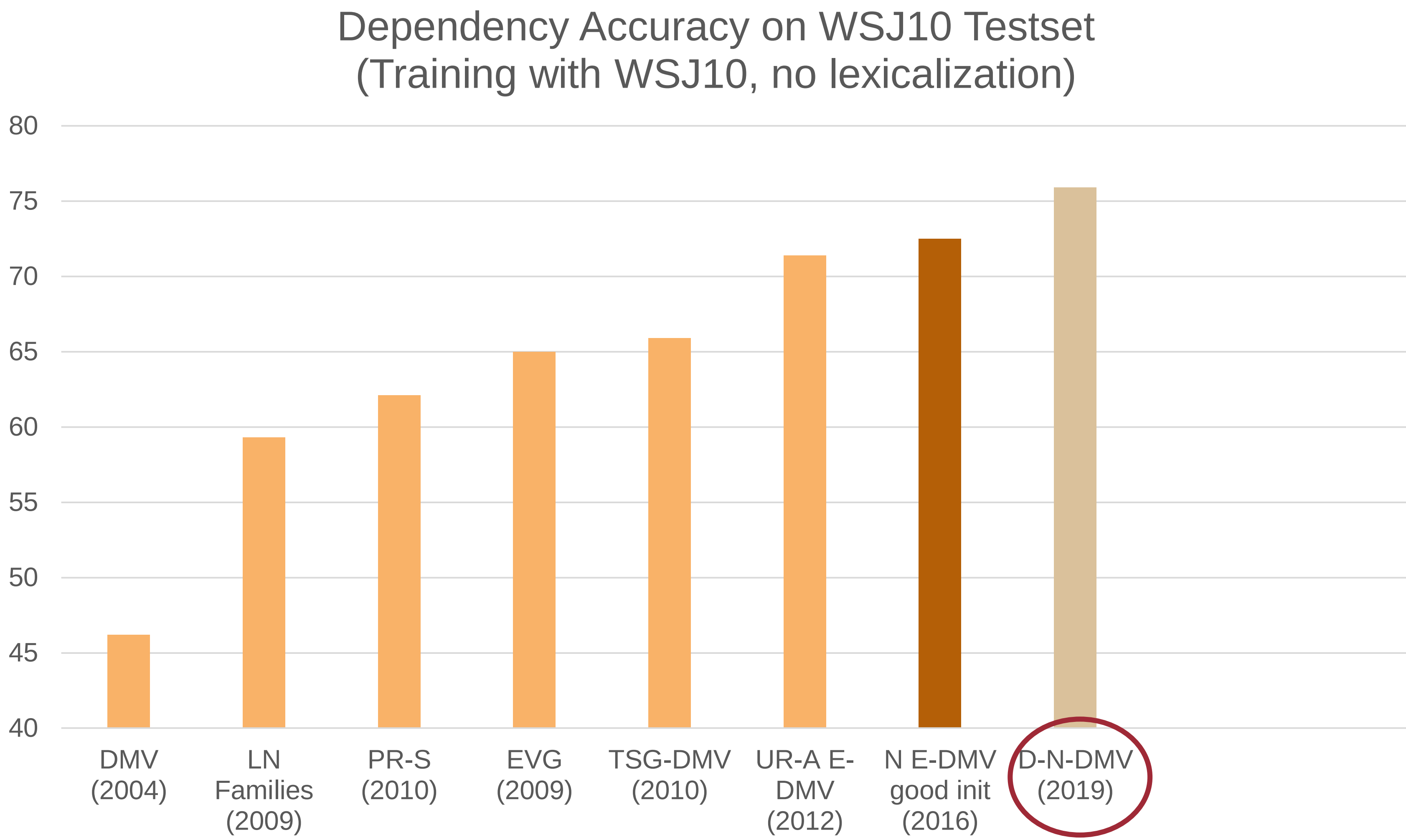
Also applicable to constituency parsing [Kim et al., "Compound PCFG for Grammar Induction", ACL 2019]



N-DMV objective: $J(\theta) = \sum_y P_\theta(x, y)$

D-N-DMV objective: $J(\theta) = \sum_y P_\theta(x, y|v_w)$

Experimental results



Generative Approaches

Discriminative Approaches

Part 2 Summary

- Neural approaches to grammar induction
  - Unfold inference as neural networks
    - Easy to implement, parallelize, scale up
  - Symbol embedding and neural parameterization
    - Captures similarity & correlation between symbols
    - Informed smoothing
  - Contextualize grammar rules
    - Break the context-free assumption, more expressive

Summary

Summary

- Symbolism vs. Connectionism
  - Each has its own pros and cons
  - Even in the era of deep learning, symbolic approaches should not be ignored
- Integrating symbolism & connectionism is a fruitful direction
  - Turning symbolic systems to neural networks
  - Learning symbolic systems using neural networks

Thank you!

Q&A